# Interactive Data Base Management System

# for Ecological Studies Related to

# the Donald C. Cook Nuclear Power Plant

by

William Y. B. Chang

and

Maryam S. Shahraray

Under Contract With

American Electric Power Service Corporation

Indiana & Michigan Electric Company

Special Report No. 119

of the

Great Lakes Research Division
Great Lakes and Marine Waters Center
The University of Michigan
2200 Bonisteel Blvd.
Ann Arbor, MI 48109

1986

ACKNOWLEDGMENTS

We would like to acknowledge the cooperation and assistance obtained from all the Principal Investigators of the D. C. Cook Nuclear Power Plant water quality studies during various phases of this project, and are particularly indebted to Dr. Ronald Rossmann for his continuous support and insightful commentaries during the course of this project and for his critical review of this document. Assistance rendered by Michael Winnell, James Barres, and David Bimber is gratefully acknowledged. We thank Drs. James Bower and David J. Jude for reviewing this document and for providing valuable suggestions.

TABLE OF CONTENTS



(continued)

TABLE OF CONTENTS (continued)



(continued)

TABLE OF CONTENTS (concluded)

LIST OF TABLES

## LIST OF FIGURES

# INTRODUCTION

Ecological studies related to the Donald C. Cook Nuclear Plant are unique. They differ from most environmental impact research in the extent of their coverage and the great length of the research period. This investigation included studies of bacteria, phytoplankton ecology, zooplankton ecology, benthic macroinvertebrate populations, fisheries, water chemistry, and physical limnology (winds, currents, ice, sediments). This study can serve as a valuable example for future environmental monitoring and is a great source of information for the enhancement of our understanding of the long-term dynamics in the nearshore region of a large lake. Accumulated data can be used for future power plant site planning and siting on the Great Lakes coastline.

The data base was begun in 1970 and has since grown rapidly in size. Although data archiving has been maintained continuously by each section, the archived information does not interface easily and cannot be used readily by a person without computer training. Furthermore, due to the rapid expansion of the data base, few individuals can maintain complete awareness of all available data and utilize pertinent information when analyzing a problem. The result is inefficiency in report writing and data interpretation. To improve this situation, a project to devise an interactive data base management system was initiated. This data system can improve efficiency in data retrieval and storage and house in one place all information from the studies related to the D. C. Cook Nuclear Power Plant. The latter provides a safeguard for access to these data by other interested persons for their research on Great Lakes ecosystems after the completion of the project. This data base encompasses 13 individual studies and is summarized in Table 1. The total includes more than

TABLE 1. Summary of data sets.

| Sub-project | Sample Type | Years Archived |
|---|---|---|
| Phytoplankton | Lake Samples | 1970-1982 |
| | Entrainment Samples | 1975-1982 |
| Zooplankton | Lake Samples | 1970-1982 |
| | Entrainment Samples | 1975-1982 |
| Benthos | Lake Samples | 1970-1982 |
| | Entrainment Samples | 1975-1982 |
| | Impingement Samples | 1975-1981 |
| Field-caught Fish | Summary Statistics | 1973-1982 |
| | Lake Samples | 1973-1982 |
| | Impingement Samples | 1973-1982 |
| Larval Fish | Lake Samples | 1973-1982 |
| | Entrainment Samples | 1975-1982 |
| Nutrient and Anion | Entrainment and Lake Samples | 1974-1982 |
| Lake Water Chemistry | Lake Samples | 1973-1982 |
| Sediments | Lake Samples | 1973-1977 |

one-half million cases of biological, chemical, and physical information on the nearshore of southeastern Lake Michigan.

The data base management program used is Taxir, which is an information storage and retrieval program and has general purpose features for keeping track of large organized data sets. The capability and benefits of such a system extend far beyond this and include the following items:

1. It offers a centralized data source and can be controlled and monitored effectively.
2. It allows for multiple users.
3. It reduces the data redundancy and increases efficiency in data retrieval and storage.
4. It can enhance data integrity, consistency, and accuracy and can provide security while greatly reducing efforts needed for generating reports and tables.

For this study, a general procedure was followed. The first step was to reorganize "raw data" in a form compatible with the Taxir data program. Comments were then solicited from each sub-project leader with respect to the appropriateness and adequacy of the information. If important parameters should be included but were not yet included in the data base, efforts were made to incorporate such information. Data verification was then performed, and errors were corrected. Because it was our intent that the data base be accessible to all interested persons, an interactive user program was written which was designed to improve user access and ease of operation.

This report documents in detail the procedures used and the programs written for establishing the data base management system for the D. C. Cook ecological study and describes the data contained in the data base as well as the ways

in which these data can be accessed. An overview of each chapter is given below.

Chapter 1 is a review of Data Base Management Systems (DBMS) and their importance in scientific research, as well as the elements that need to be considered when selecting a DBMS program.

Chapter 2 is an introduction to available (DBMS) programs from the Michigan Terminal System (MTS) of The University of Michigan along with the characteristics of each DBMS program. Comparisons between these systems are made. The reasons for choosing the Taxir program for this task are discussed.

Chapter 3 is an introduction to data organization and how Taxir handles data structure. It also describes major terms and notations used in association with the establishment of the data base.

Chapter 4 is a description of the procedures and the flow diagram for establishing a data base for each sub-project. Details are given regarding the structures of different data sets, which include formats, parameters, ranges, software programs, and methods. All procedures and software programs used are described and documented. Explanations of how to access the data sets are also given.

Chapter 5 includes a presentation of the Interactive Computer Program, which is written in FORTRAN and can be interfaced with Taxir. This program is intended to improve user access and can help persons with little experience in accessing data with computers.

# CHAPTER 1

# DATA BASE MANAGEMENT SYSTEMS

Data Base Management Systems (DBMS) is a term that refers to the computer technology necessary for data collection, organization, storage, retrieval, and manipulation. A data base system (DBS) deals with record-keeping and making the computer function as a super filing system.

A Data Management System (DMS) works with the smallest unit of data, which is usually called a data item. A collection of items is called a record, and an organized collection of records constitutes a file. The occurrences of the records in a file are associated by means of a specified relationship.

The generally-accepted requirements for a DBMS include the capability to create, revise, add, and delete records from a file; retrieve records; sort; perform limited computations; and generate reports.

## ADVANTAGES OF USING A DBMS

The advantages of working with a DBMS rather than a non-computer filing system are that the data can be accessed in a greater variety of ways, searched quickly, and changed more easily; and auxiliary programs can be applied to the data in the DBMS to produce reports, copies of the data, and other outputs. In non-computer filing systems, such outputs usually must be produced manually, a process which is time-consuming, subject to transcribing errors, and inefficient.

The advantages of Data Base (DB) technology in areas other than data filing can include:

- data independence,
- data shareability,
- non-redundancy of stored data,
- reliability,
- integrity,
- access flexibility,
- security,
- efficient performance (especially in view of large-size files), and
- greater administrative control.

These advantages are essential for developing and supporting modern integrated information systems, which bring together a variety of data and interrelate it for a variety of users, not just for one or for a limited few. The ability to share or use these data also minimizes the amount of redundant storage.

## DATA BASE DESIGN

Data base design is the most important attribute of a data base management system and shows how the data items are classified and interrelated. In designing non-computer filing systems, an office-clerical typically invents a number of filing categories and decides which files are in which categories and what kinds of things are sorted in each file. The same design decisions must be made when computer systems analysts set up data bases.

## TYPES OF COMPUTER DATA BASE MODELS

Technically, data base designs can be grouped according to their logical characteristics. One data base design may have many file categories with files within each category, another may allow each file to belong to several categories at once, and so on.

Each DBMS has certain capabilities and restrictions regarding the data base designs that it will support. The set of rules that determines which data base designs a given DBMS can support is called the "data model" of the DBMS. The term "data model" is used because the data base design rules for each DBMS constitute a set of assumptions about how a filing system generally behaves.

Because most data base management systems are expected to be general-purpose computing tools, it is the goal of most data models to be as general and as simple as possible. There are three major kinds of data models that are common: hierarchical, network, and relational. Of these, only the relational model maintains the logical simplicity of so-called flat files. A brief description of these models follows.

### Hierarchical Model

The hierarchical model keeps data in a form resembling the following outline. For example, suppose there is a need to keep records of university students who are being used as subjects in psychology experiments. A hierarchical system would organize the data like this:

| | | |
|---|---|---|
| I. | Researcher<br>Address<br>Experiment Type | :Dr. Know<br>:Medical Bldg.<br>:Attention |
| | A. Subject<br>Telephone<br>Major<br>Sex | :John<br>:665-9988<br>:Computer Science<br>:Male |
| | B. | : |
| | C. | : |
| II. | Researcher | : |

The basic idea of the hierarchical model is that the data are organized into groups (in our example, researchers) which have subgroups (individual subjects in this instance); in turn, they can have subgroups, and so on, to the required degree of complexity.

It is important to note that each group in a hierarchical data base may have many subgroups, but each subgroup is in one and only one group. This is an example of what is called a "1-N" relation between groups and subgroups.

The major failing of this model is that few data base management problems remain strictly hierarchical. For example, if one person serves as a subject for two different reseachers, then the data would no longer behave in a strictly hierarchical manner.

Network Model

The network model could be considered an extension of the hierarchical data model. A network data model is simply a model that allows more or less arbitrarily complex relationships. Thus, in our example not only can the records of one researcher be related to those of many subjects, but also one

subject record can be related to many researcher records. This is called a many-many relationship. The following relationships can serve as an example:

| Researcher | | | Researcher |
|---|---|---|---|
| Dr. Know | | | Dr. Best |

| Subject | Subject | Subject | Subject |
|---|---|---|---|
| John | Mary | Bob | Nancy |

The biggest problem with the network data model is that the data base can become excessively complicated.

## Relational Data Model

The relational data model is the most modern of all the data models. This is actually an extension of the flat-file data model (see Chapter 3). The flat-file data model treats data as a single collection of ordered items, each with the same format. For example, the following flat file contains the information about the subjects in our previous example,

| Subject | Telephone | Major | Sex |
|---|---|---|---|
| John | 665-9988 | Computer Science | Male |
| ' | ' | ' | ' |
| ' | ' | ' | ' |
| ' | ' | ' | ' |

while the following flat file keeps track of researchers:

| Researcher | Address | Experiment Type |
|---|---|---|
| Dr. Know | Medical Bldg. | Attention |
| ' | ' | ' |
| ' | ' | ' |
| ' | ' | ' |

Then if we need to store the information about the relationship between researchers and subjects, there will be another flat file such as the one below:

| Researcher | Subject | Date Assigned |
|---|---|---|
| Dr. Know | John | 4/1/82 |
| ' | ' | ' |
| ' | ' | ' |
| ' | ' | ' |

In order to have a practical records system using these flat files, we would need some data base management capable not only of storing them but also of manipulating them to retrieve and update the stored information. This data base management system would have to extract and combine information from the flat files in various ways, but the relational model has a flexibility that the flat-file data model cannot attain.

In general, any data base design that can be represented in the hierarchical or the network models can also be represented as a set of relations in the relational model. While it is true that the relational model is conceptually simple, there still remains a good deal of work to make that simplicity available to users at low cost.

AVAILABILITY OF DBM

The commercial products associated with data bases and data management systems have changed dramatically in the last several years, especially because of new hardware and software technologies. In the 1970s there were no more than two dozen widely marketed DBMS product lines; today there are hundreds. Major data management systems differ primarily in language utilized, file organization, and access method and level of system users.

## Language Utilized

This deals with the difference between what are called self-contained and host-contained systems. The former usually provide a language designed for the non-programmers, whereas the latter are tied to such languages as COBOL, PL/1, ALGOL, and FORTRAN, and are especially for the application programmers who use these languages. Self-contained systems provide their own language, which is user-oriented and designed to be used by managers and others with minimal knowledge of programming.

It is important to note that self-contained languages are usually machine dependent. Few systems can operate effectively on more than one type of computer equipment.

## File Organization

In examining data management systems, one must separate logical and physical file organization. A study of logical organization will show whether user requirements can be satisfied. A study of physical file organization indicates the handling procedures, file maintenance, retrieval capability, output, and similar operational features of the data management system.

## Access Method and Level of System Users

A data management system makes it possible for various users to work with a common data base when the data base is an interrelated set of organization files. Three levels can be defined within the system users:

1. Systems Designers determine long-range objectives.
2. Middle Management oversees the daily operations of a company.
3. General Users need to have no knowledge of programming, and/or self-contained system language.

## CRITERIA FOR SELECTING A DBMS

Too often, decisions on the selection of a DBMS are based on incomplete and inaccurate factors that can result in revisions and costly long-range repercussions. Selection of a DBMS should be done carefully in order to avoid a loss of time and money in the long run. In selecting a data base management system, four criteria should be considered:

1. suitability of the DBMS to the specific characteristics of the data to be handled,
2. simplicity of the system used,
3. time and cost involved, and
4. future needs.

A good DBMS should offer: 1) data independence, 2) data dictionary to define and control the data environment, 3) good query language to allow access to the data base, 4) good report-generating features, and 5) a simple high-level user language.

The time dedicated to an analysis and evaluation of the user's requirements and limitations, as well as the evaluations of the assets and limitations of the various DBMS available, can be the time most valuably spent. The planning time, carefully spent, can make the installation and use of a DBMS a profitable experience in both financial and managerial terms.

# CHAPTER 2

# DBMS SUPPORTED BY MTS

## MICHIGAN TERMINAL SYSTEM

The Michigan Terminal System (MTS) is a very powerful operating system supported by the University of Michigan Computing Center. Like most large computer installations, it prohibits direct operation of the equipment by anyone but professional operators. The computer is under the combined control of its human operators and a master program called the operating system, which coordinates the jobs of the various users and provides them with a variety of auxiliary computing services.

MTS permits its users to operate in either interactive mode or batch mode. In interactive mode, a large number of users at remote terminals are able to use the computer concurrently and independently. The user in interactive mode usually types a message (e.g., command, statement, query) on his terminal keyboard and sees a response from the computer before typing the next statement, which may be a modification of the first one.

For jobs running in batch mode, the operating system is the same, but no dialogue is possible between the user and the computer. The user submits his entire job at once (in the form of a card deck) and waits for the entire output of the job. Batch mode is useful when there is no need for human-computer interaction. In both modes, the user appears, from his own point of view, to have the entire computer to himself.

## DBMS ON MTS SUPPORTED BY THE COMPUTING CENTER

The computing center supports the Taxir and SPIRES data base management systems. Both systems are used primarily for academic applications, although they have been applied to a few administrative projects.

### Taxir

Taxir is a generalized information storage and retrieval system which can be used at any computer installation within the MTS operating system. It is written and supported by the University of Michigan Computing Center. It is a completely self-contained system that provides facilities for defining, searching, and managing data bases that can be implemented as flat-file data bases.

Taxir is the simplest data base management system available on MTS at The University of Michigan and is the easiest system to learn and use. In general, Taxir is very inexpensive to use. It stores data in a highly compressed form, which reduces the cost of storage; and it retrieves data very rapidly, thereby reducing computer processing costs.

Taxir can be run in both interactive and batch mode. It provides a number of features that make it an attractive data base management system for many data base applications. It has a single high-level language, somewhat resembling English, that provides simple commands for defining, manipulating, updating, and querying data bases. The system has flexible report-generation facilities, which allow users to produce ordered, labeled, formatted outputs. Taxir provides an interface with MIDAS, a powerful statistical analysis program available to compensate for Taxir's few facilities for statistical features. In addition, it can be called through standard FORTRAN-calling conventions. In general,

Taxir can best be used for data base applications in which 1) the data are represented in a flat-file structure, 2) the application requires a cheap, easy-to-use system, 3) users want to search on any field or combination of fields, and 4) MTS security facilities are sufficient.

SPIRES

The Stanford Public Information Retrieval System (SPIRES) was written at Stanford University. It is an on-line general-purpose information and retrieval system available in the MTS operating system and supported by the Computing Center at The University of Michigan. It is a completely self-contained DBMS that provides extensive facilities for defining, searching, updating, and managing data bases. SPIRES is based on the hierarchical data model, but it also provides some network capabilities (see Chapter 1).

Although SPIRES is a general-purpose system, it has special features for efficiently and conveniently storing and retrieving text or character data. Any application that requires storage of lengthy textual material is a strong candidate for SPIRES. For example, designers of bibliographic data bases would probably find SPIRES the most appropriate data base management system on MTS.

Output formats and other special SPIRES features can be used to generate reports, construct tables, sort data, and display data in a variety of formats. Because SPIRES is so flexible, it is also very complex. This means that it is more costly and often more difficult to use than the other DBMS available on MTS. It is true that searching an existing SPIRES data base is not difficult, but defining a new data base usually is not a simple task. Some programming background is often required for a better understanding of this language.

In general, one should consider SPIRES for data bases when the data 1) involve lengthy textual materials such as characters or strings, 2) require many

repeating fields, 3) can best be stored hierarchically, 4) are very large in number, and 5) require extensive DBM facilities.

OTHER DBMS

Other units at The University of Michigan also support some DBMS on MTS, which may be used by anyone with a computing center account. The major ones are MICRO, MIDAS, OSIRIS, ADBMS, ARCH:MODEL, and Relational Management System (RIM). A brief description of each system follows.

MICRO

The MICRO information management system, which operates only on MTS, is written and supported by the Institute of Labor and Industrial Relations, a joint institute of The University of Michigan and Wayne State University. MICRO is a self-contained information, storage, and retrieval system. It is based on the relational data model.

MICRO permits non-programmers to define, enter, interrogate, manipulate, and update user-defined collections of data in a relatively unstructured and unconstrained environment. It has general applicability to a wide variety of educational, administrative, and research data processing activities. Its capabilities lie roughly between those of Taxir and SPIRES. It is designed to be run interactively from computer terminals, but caution is required when trying to run a batch job.

MICRO is very powerful in terms of the programming language because of its English-like grammer, which makes it easy to learn and use. Predetermined procedures can be easily executed in MICRO to deal with complex reporting and retrieval problems. It has limited facilities for character string (text)

data. It can be interfaced to MIDAS (the statistical package on MTS) for any statistical analysis.

## MIDAS and OSIRIS

These are both statistical analysis packages which provide some data management capabilities. MIDAS is supported by the Statistical Research Laboratory. OSIRIS is supported by the Survey Research Center of the Institute for Social Research. They both work in interactive and batch modes.

## ADBMS

This is a host-language-dependent DBMS based on the network data model. Users must write their own "interface" program to use ADBMS. It is written in FORTRAN but can be called from programs written in FORTRAN, COBOL, and PL/1 on MTS. ADBMS is particularly useful when one is faced with complex structures and when access and reporting requirements are algorithmic in nature. It also can be used with many different operating systems and computers.

ADBMS is written and supported by a Research Project of the Information System Design Optimization Society (ISDOS) in the department of Industrial and Operations Engineering (IOE), the College of Engineering at The University of Michigan.

## ARCH:MODEL

ARCH:MODEL is a DBMS designed for applications involving geometric modeling, such as computer-aided architectural design. It is supported by the Architectural Research Laboratory of the College of Architecture and Urban Planning at The University of Michigan.

Relational Management System (RIM)

This is a self-contained system based on the relational model. It provides features for combining and manipulating flat files. Data can include real and integer vectors and also matrices. It has extensive on-line documentation and help facilities. It is supported by the Architectural Research Laboratory but is available to all in the University community.

## CRITERIA FOR A SUITABLE SELF-ORIENTED DBMS

In choosing a good self-contained high-level language, one should consider the following properties:

1) A substantial number of prospective users of the language must exist;
2) The language must solve a substantial portion of the problems confronting the intended users;
3) It should not be needlessly difficult to learn;
4) It should be natural to write programs in the language which are easy to understand;
5) Any limitation of the language should be clearly justified (e.g., learning ease, processing efficiency, available capacity); and
6) The language should provide the users with appropriate access to facilities for effective communication with the environment.

## JUSTIFICATION FOR SELECTION OF TAXIR

After careful study and comparison of the available DBMS on MTS, Taxir was chosen for this project. Comparisons were done mainly between Taxir, SPIRES, and MICRO. All three are general-purpose DBMS and are simpler

and stronger than the others. Although SPIRES has many good features, running SPIRES programs is costly and time consuming. Furthermore, the advantage which SPIRES has in dealing with bibliographic features is not of interest here. Because of the complexity involved in the design of a new SPIRES data base, it has been recommended that, when possible, one should use Taxir or MICRO instead of SPIRES.

Because in this study data sets are represented in forms of flat files, the final comparisons were done between MICRO and Taxir. Although MICRO can work with several data sets simultaneously and has rather good on-line documentation, Taxir has the following advantages:

1) Taxir is written, supported, and maintained by the Computing Center of The University of Michigan;
2) Taxir is the simplest DBMS to learn and use;
3) It is cheaper to run a Taxir program;
4) Taxir stores data in a highly compressed form (less memory space is needed);
5) Information retrieval is faster with Taxir;
6) Taxir has flexible querying and display features;
7) It is possible to call Taxir from a FORTRAN program for further applications; and
8) Taxir can be run safely in both interactive and batch modes.

These advantages determined the choice of Taxir for the DBMS for the D. C. Cook environmental impact study.

CHAPTER 3

# TAXIR ORGANIZATION

## TAXIR DATA BASE

A Taxir data bank is a collection of items, each of which contains data belonging to information categories called descriptors containing all the relevant information about some entity being described by the data bank. For example, a phytoplankton data bank would contain one item for each species on the list. Each item in that data bank would consist of several pieces of information about one presented species, such as day, month, year, location, major group, cell number, and so on.

Each descriptor represents an attribute of the entities being described by the data bank. For example, a phytoplankton data bank could have the descriptors such as day, month, year, location, major group, cell numbers, etc. That is, each item in a data bank is associated with a series of data values (descriptors). There should be one value for each descriptor. Thus a Taxir data bank may be thought of as a two-dimensional matrix in which each row corresponds to an item and each column corresponds to a descriptor. This data organization is called a flat-file structure (discussed in next section). Every Taxir data bank has the following structure:

```
         Descriptor 1    Descriptor 2    Descriptor 3...........Descriptor N

Item 1       '\              '\              '\                     '\
Item 2       ' |             ' |             ' |                    ' |
   '         ' |             ' |             ' |                    ' |
   '         ' } States      ' } States      ' } States             ' } States
   '         ' |             ' |             ' |                    ' |
   '         ' |             ' |             ' |                    ' |
Item M       '/              '/              '/                     '/
```

The range of values allowed for a descriptor in a data bank is called a descriptor state, which may be integer/real numbers or strings of characters. This concept is identical to that of range for a parameter. For example, if the descriptor state for years is 1971 to 1973, then the descriptor year could have the states 1971, 1972, and 1973.

## FLAT-FILE DATA MODEL

As was mentioned in Chapter 1, the relational model is the extension of the flat-file data model. The flat-file data model is the simplest and the oldest data model. A flat-file DBMS keeps data in the form of a flat file, (e.g., mailing list data bank). Each item in the flat file is called a record. Each record corresponds to a single complete entry in the file. Records are composed of data elements.

A data element is basically an irreducible data component. Each data element has a name or a value. Data elements are sometimes called fields. Every record in the flat-file data base has the same number of elements, and each record has data values that represent one object in the real world.

## FLAT-FILE NATURE OF TAXIR DATA BANK

The way in which Taxir organizes and manipulates data is based on a simple notion of set theory using a flat-file data model. In general, if the information can be stored as a single flat file, then the data base can be stored as a Taxir data bank. In each Taxir data bank, each item has one and only one state for each descriptor. That is, each item in a Taxir data bank corresponds to a record in a flat-file data base, each descriptor corresponds to a field,

and each descriptor-state corresponds to a value. Furthermore, like other flat-file DBMS, Taxir does not allow structure fields (i.e., descriptors may not consist of other descriptors); and it provides no direct access to the items in one data bank based on information stored in another data bank.

## DESIGN OF THE DATA BANK

Before using Taxir, one needs to consider the following for the design of a data bank:

1) Number of data banks needed
2) Item(s) in each data bank
3) List of the descriptors for each item
4) List of states for each descriptor
5) Kind of queries (questions) expected
6) Nature of information flow and work flow
7) Cost and time involved

In most cases, planning and time will be needed to decide upon these points, but it is essential to study all the constraints in order to design and create a data bank that meets all the requirements.

## TYPE OF DATA SUPPORTED BY TAXIR

Associated with each descriptor in a Taxir data bank is a descriptor type which specifies what data values may be used as states for that descriptor. Taxir permits three descriptor types: from-to, order, and name, which provide the capability to store numerical (integer/real values), codified (categorical), and general character-string (words) data, respectively. Taxir also provides

some features for handling missing data for each of these descriptor types identified as unknown states. The descriptor types are specified by the designer of a data bank when the data bank is defined.

The thing to consider here is that Taxir automatically assigns code numbers for both descriptors and their states to a data bank. Thus a user can have the choice of typing either the name of the descriptors and their states or the codes for both of them when communicating with Taxir.

## RUNNING TAXIR

There are three ways to use Taxir: 1) as an interactive system from a terminal; 2) as a non-interactive system in batch mode, and 3) as a subroutine by calling it from a user program. In each case, user inputs to Taxir must be in the form of Taxir statements, MTS commands, or input data. Each Taxir statement is a request for Taxir to do some data processing or to modify the Taxir environment. Each statement begins with a statement name, known as a statement type. Most statements also include additional information, which further specifies what the user wants Taxir to do. The system reads and executes each statement before treating the next statement. There are no facilities for conditional jumps; hence, no program branching or looping. In other words, the execution of the Taxir statements is sequential.

## DATA ENTRY AND DATA COMPRESSION

Taxir provides the following data entry capabilities:

1) Data can be entered in batch mode or in interactive mode.

2) Data can be entered directly from punch cards, MTS disk files, or terminals, and indirectly from magnetic tapes and other available machine-readable forms.

3) Data to be entered can be in a variety of user-specified fixed or free formats.

4) Taxir assigns a default state for any missing values.

5) Taxir does some data validation on all data.

6) Data are automatically stored in a highly compressed form.

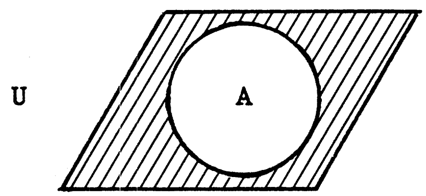
This last capability of Taxir is particularly important in saving memory space. The compression process is completely automatic and unseen by Taxir users who cannot (need not) control it.

## RETRIEVING DATA AND BOOLEAN EXPRESSIONS

The power to retrieve information selectively from a data bank is the power to name subsets of interest from the total bank. For this purpose, Taxir applies the language of boolean algebra.

A boolean expression which defines the subset of items from the data bank consists of a series of operators and operands. A set of rules defines how the operators act on the operands to yield a result. These operators are complement (NOT), intersection (AND), and union (OR).

The operands are sets, as are the results. The figures below explain these concepts graphically:



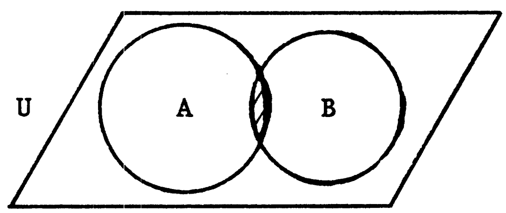

(1)

Complement,

if U = all the students in the class and

A = those with hats,

then NOT A = those without hats (shaded area).




(2)

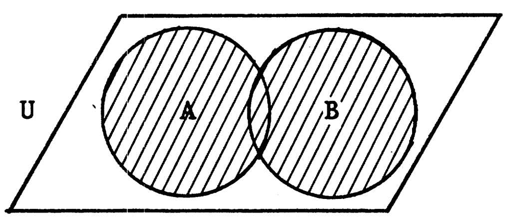
Intersection,

if U = all the students in the class and

A = those with hats and

B = those with coats,

then A AND B = those with both hats and coats (shaded area).




(3)

Union,

if U = all the students in the class and

A = those with hats and

B = those with coats,

then A OR B = those with hats or coats or both (shaded area).

Taxir boolean expressions provide simple, flexible ways for users to select items by specifying simple or complex search criteria involving any or all of the descriptors in a data bank. The retrieving statements enable Taxir users to:

1) Find out how many items meet some user specified criteria, prior to displaying, deleting, or updating them.
2) Retrieve and immediately display some user-specified set of items.
3) Retrieve items based on the states of any descriptor in a data bank.
4) Issue complex search requests involving any combination of descriptors.
5) Narrow a search result before displaying or updating the items in it.

## DISPLAYING DATA IN TAXIR

Taxir display facilities include features for:

1) Displaying some or all of the descriptor-state for selected items.
2) Displaying data in a variety of formats.
3) Sorting output in terms of any descriptor.
4) Generating a report, including subtotals and totals.

(Note that Taxir can interface with MIDAS for further statistical computations.)

# CHAPTER 4

# PROGRAMMING PROCEDURES FOR ESTABLISHING THE COOK DATA BASE

## DESCRIPTION OF THE COOK DATA BASE

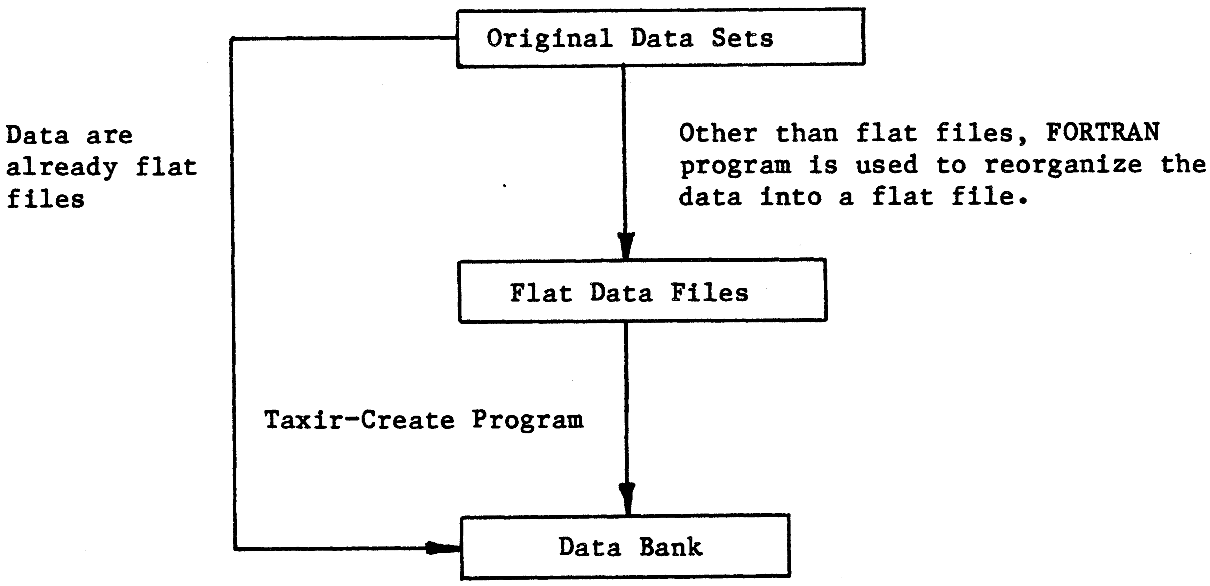
The methods used to establish the Cook Project data base are discussed in this chapter. This data base includes 15 data banks which are shown below:

| Group | Data Type | Name of Data Bank |
|---|---|---|
| Phytoplankton | Lake Samples | 1. Lake.Phytoplankton |
| | Entrainment Samples | 2. Entrained.Phytoplankton |
| Zooplankton | Lake Samples | 3. Lake.Zooplankton |
| | Entrainment Samples | 4. Entrained.Zooplankton |
| Benthos | Lake Samples | 5. Lake.Benthos |
| | Entrainment Samples | 6. Entrained.Benthos |
| | Impingement Samples | 7. Impinged.Benthos |
| Field-Caught Fish | Summary Statistics | 8. Adult.Fish.Summary.Statistics |
| | Lake Samples | 9. Lake.Adult.Fish |
| | Impingement Samples | 10. Impinged.Adult.Fish |
| Larval Fish | Lake Samples | 11. Lake.Larvae |
| | Entrainment Samples | 12. Entrained.Larvae |
| Nutrient and Anion | Lake and Entrainment Samples | 13. Nutrients |
| Lake Water Chemistry | Lake Samples | 14. Lakewater |
| Sediments | Lake Samples | 15. Sediments |

The explanations for the methods used along with the lists of the programs, the examples, and other helpful comments are provided in the sections which follow. It is hoped that this information can facilitate users in accessing data of interest, retrieving the needed information, and using the provided methods for establishing a similar data base in the future.

GENERAL PROCEDURES

To establish a data base using Taxir, the data must be in flat-file form. Once they are in that form, a Taxir program is used to enter these data to the computer data base program. Because most of the Cook data were not in the form of flat files, FORTRAN programs were written to reorganize these data into the flat-file forms. The general procedures involved in this operation are shown in the following flow chart.




For each data set, the FORTRAN program and the programs to create a Taxir data base for this data set are presented along with the flow chart diagram. It is noted that the FORTRAN programs written for this project are used not only to reformat the data sets but also in some cases to combine the parameters of interest from several data sets into a single flat file.

LAKE PHYTOPLANKTON

The Lake.Phytoplankton data bank is one of the largest data banks created for this project. It contains 13 descriptors and 90,076 items. The descriptors and their codes are listed below:

1-DAY

2-MONTH

3-YEAR

4-LOCATION

5-NAME

6-TEMPERATURE

7-SPECIES CODES

8-MAJOR GROUPS

9-CELLS

10-FRACTION

11-TOTAL CELLS

12-DIVERSITY

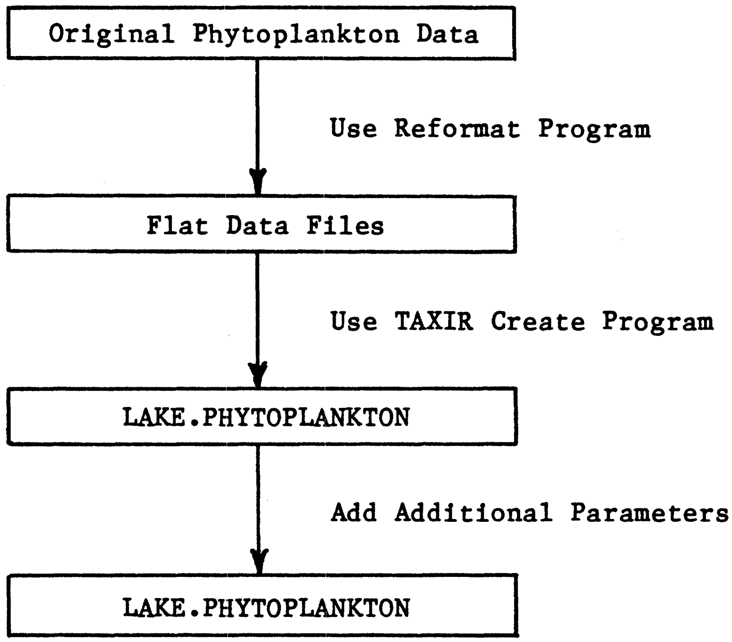
13-REDUNDANCY

The monthly number of phytoplankton items collected since November 1970 is listed in Table 4.1.

TABLE 4.1. The number of data items in the Lake.Phytoplankton data bank for the D. C. Cook Plant data.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 70 | NOV | (1,179) | 76 | APR | (2,105) | 79 | APR | (2,302) |
| 71 | APR | ( 988) | | MAY | ( 735) | | MAY | ( 674) |
| | JUL | (1,799) | | JUN | ( 545) | | JUN | ( 736) |
| | SEP | (1,589) | | JUL | (1,925) | | JUL | (1,516) |
| | NOV | ( 985) | | AUG | ( 504) | | AUG | ( 690) |
| 72 | APR | (1,078) | | SEP | ( 880) | | SEP | ( 491) |
| | JUL | ( 758) | | OCT | (2,441) | | OCT | (2,045) |
| | OCT | (1,607) | 77 | APR | (2,405) | | NOV | ( 724) |
| 73 | APR | (1,621) | | MAY | ( 602) | 80 | APR | (2,108) |
| | JUL | (1,321) | | JUN | ( 598) | | MAY | ( 696) |
| | OCT | (1,495) | | JUL | (1,861) | | JUN | ( 653) |
| 74 | APR | (1,628) | | AUG | ( 708) | | JUL | (1,919) |
| | MAY | ( 513) | | SEP | ( 991) | | AUG | ( 850) |
| | JUN | ( 498) | | OCT | (2,390) | | SEP | ( 725) |
| | JUL | (1,654) | | NOV | ( 590) | | OCT | (2,180) |
| | AUG | ( 393) | 78 | APR | (2,432) | | NOV | ( 555) |
| | SEP | ( 338) | | MAY | ( 852) | 81 | APR | (1,705) |
| | OCT | (2,190) | | JUN | ( 989) | | MAY | ( 695) |
| 75 | APR | (1,727) | | JUL | (2,897) | | JUN | ( 371) |
| | MAY | ( 393) | | AUG | ( 638) | | JUL | (1,248) |
| | JUN | ( 685) | | SEP | ( 671) | | AUG | ( 571) |
| | JUL | (1,461) | | OCT | (2,610) | | SEP | ( 762) |
| | AUG | ( 455) | | NOV | ( 743) | | OCT | (2,080) |
| | SEP | ( 643) | | | | | NOV | ( 655) |
| | OCT | (2,839) | | | | 82 | APR | (1,749) |
| | | | | | | | MAY | ( 429) |

The flow chart representing the stages in the creation of the Lake.Phytoplankton data bank is as follows:




<u>Phytoplankton Data Files and the Reformat Programs</u>

The original phytoplankton data files are stored in the files: RDmonthyear (for example, RDNOV70). The data between November 1970 and December 1980 are saved on the tape "Phyto," the 1981 and 1982 data files can be found on the "Phyto2" tape. Because there are some differences in the data structures in these files, especially in the formats of the headlines, different reformat programs are needed to handle these forms of the data structures. The following is a list of the names of the reformat programs for different sets of the data.

| Reformat Program | Data Set |
|---|---|
| REFORMAT1 | All months of 1971, 1972, 1973; and May, June, August, September in 1974. |
| REFORMAT2 | November 1970; all months of 1975; and April, May, July, October in 1976. |
| REFORMAT3 | April, July, October in 1974. |
| REFORMAT4 | June, August, September in 1976; and all months of 1977, 1978, 1979, 1980, 1981, and 1982. |

These reformat programs are listed in Tables 4.2-4.5.

## Taxir Create Program

Reformat programs discussed in the last section were used to provide the phytoplankton data in the form of flat files. The Taxir Create program PLTAXIRCR was then used to create the LAKE.PHYTOPLANKTON data bank. The contents of the PLTAXIRCR program are given in Table 4.6.

TABLE 4.2. Program REFORMAT1.

```
C PROGRAM  REFORMAT1  IS RUN TO REORGANIZE LAKE PHYTOPLANKTON RAW DATA
C FILES.  THIS PROGRAM IS USED FOR ALL MONTHS OF 1971, 1972, 1973; AND
C MAY, JUNE, AUGUST, SEPTEMBER IN 1974.
C UNIT 5 IS ASSIGNED TO INPUT FILE. (RAW DATA FILE).
C UNIT 6 IS ASSIGNED TO OUTPUT FILE.·(REFORMATTED DATA FILE).
C UNIT 7 IS ASSIGNED TO ANOTHER OUTPUT FILE FOR THE VALUES OF DL & SW.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      LOGICAL*1 CODE(9,200)
      REAL CTS(200),FRAC(200),H(16)
C
      CONST=-1./ALOG(2.)
C
C*******
C*******        READ THE HEADLINES.
C*******
C
 1    READ(5,100,END=99) (H(I),I=1,16),IDL,ISW
 100  FORMAT(16A4,I2,1X,I3)
      WRITE(7,200) IDL,ISW
 200  FORMAT(I2,1X,I3)
C
      SUM=0.
      NS=1
C
C*******
C*******        READ THE DATA LINES.
C*******
C
 5    READ(5,300,END=10) (CODE(J,NS),J=1,9),CTS(NS)
 300  FORMAT(9A1,F7.0)
      IF(CTS(NS).LT.1.) GO TO 10
      CTS(NS)=CTS(NS)/1000.
      SUM=SUM+CTS(NS)
      NS=NS+1
      GO TO 5
C
 10   DIV=0.
      NS=NS-1
C
C*******
C*******        CORRECT THE FRACTION AND DIVERSITY VALUES.
C*******
C
      DO 15 I=1,NS
      FRAC(I)=CTS(I)/SUM
      DIV=DIV+FRAC(I)*ALOG(FRAC(I))*CONST
 15   FRAC(I)=FRAC(I)*100.
C
C*******
C*******        WRITE THE CORRECT VALUES.
C*******
C
      DO 20 I=1,NS
 20   WRITE(6,400) (H(K),K=2,9),H(10),(CODE(J,I),J=1,9),
     1CTS(I),FRAC(I),SUM,DIV
 400  FORMAT(8A4,6X,A4,2X,9A1,F8.1,F7.2,F8.1,F6.2)
C
      GO TO 1
C
 99   STOP
      END
```

TABLE 4.3. Program REFORMAT2.

```
C PROGRAM  REFORMAT2  IS RUN TO REORGANIZE LAKE PHYTOPLANKTON RAW DATA
C FILES.  THIS PROGRAM IS USED FOR NOVEMBER 1970; . ALL MONTHS OF 1975;
C AND APRIL, MAY, JULY, OCTOBER IN 1976.
C UNIT 5 IS ASSIGNED TO INPUT FILE, (RAW DATA FILE).
C UNIT 6 IS ASSIGNED TO OUTPUT FILE, (REFORMATTED DATA FILE).
C UNIT 7 IS ASSIGNED TO ANOTHER OUTPUT FILE FOR THE VALUES OF DL & SW.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      LOGICAL*1 CODE(9,200)
      REAL CTS(200),FRAC(200),H(16)
C
      CONST=-1./ALOG(2.)
C
C*******
C*******        READ THE HEADLINES.
C*******
C
 1    READ(5,100,END=99) (H(I),I=1,16),IDL,ISW
 100  FORMAT(16A4,I2,1X,I3)
      WRITE(7,200) IDL,ISW
 200  FORMAT(I2,1X,I3)
C
      CF=(IDL+1.)*41.4516/ISW
      SUM=0.
      NS=1
C
C*******
C*******        READ THE DATA LINES.
C*******
C
 5    READ(5,300,END=10) (CODE(J,NS),J=1,9),CTS(NS)
 300  FORMAT(9A1,F7.0)
      IF(CTS(NS).LT.1.) GO TO 10
      CTS(NS)=CTS(NS)*CF
      SUM=SUM+CTS(NS)
      NS=NS+1
      GO TO 5
C
 10   DIV=0.
      NS=NS-1
C
C*******
C*******        CORRECT THE FRACTION AND DIVERSITY VALUES.
C*******
C
      DO 15 I=1,NS
      FRAC(I)=CTS(I)/SUM
      DIV=DIV+FRAC(I)*ALOG(FRAC(I))*CONST
 15   FRAC(I)=FRAC(I)*100.
C
 *******
 *******        WRITE THE CORRECT VALUES.
 *******

      DO 20 I=1,NS
 20   WRITE(6,400) (H(K),K=2,9),H(10),(CODE(J,I),J=1,9),
     1CTS(I),FRAC(I),SUM,DIV
 400  FORMAT(8A4,6X,A4,2X,9A1,F8.1,F7.2,F8.1,F6.2)
C
      GO TO 1
C
 99   STOP
      END
```

TABLE 4.4. Program REFORMAT3.

```
C PROGRAM  REFORMAT3  IS RUN TO REORGANIZE LAKE PHYTOPLANKTON RAW DATA
C FILES. THIS PROGRAM IS USED FOR APRIL, JULY, OCTOBER IN 1974.
C UNIT 5 IS ASSIGNED TO INPUT FILE, (RAW DATA FILE).
C UNIT 6 IS ASSIGNED TO OUTPUT FILE, (REFORMATTED DATA FILE).
C UNIT 7 IS ASSIGNED TO ANOTHER OUTPUT FILE FOR THE VALUES OF DL & SW.
C
C
C
C
C*******
C*******         INITIALIZE VARIABLES.
C*******
C
      LOGICAL*1 CODE(9,200)
      REAL CTS(200),FRAC(200),H(16)
C
      CONST=-1./ALOG(2.)
C
C*******
C*******         READ THE HEADLINES.
C*******
C
 1    READ(5,100,END=99) (H(I),I=1,16),IDL,ISW
 100  FORMAT(16A4,I2,1X,I3)
      WRITE(7,200) IDL,ISW
 200  FORMAT(I2,1X,I3)
C
      IF((IDL.EQ.0).AND.(ISW.EQ.0)) CF=1
      IF((IDL.NE.0).OR.(ISW.NE.0)) CF=(IDL+1.)*41.4516/ISW
      SUM=0.
      NS=1
C
C*******
C*******         READ THE DATA LINES.
C*******
C
 5    READ(5,300,END=10) (CODE(J,NS),J=1,9),CTS(NS)
 300  FORMAT(9A1,F7.0)
      IF(CTS(NS).LT.1.) GO TO 10
      CTS(NS)=(CTS(NS)/1000.)*CF
      SUM=SUM+CTS(NS)
      NS=NS+1
      GO TO 5
C
 10   DIV=0.
      NS=NS-1
C
C*******
C*******         CORRECT THE FRACTION AND DIVERSITY VALUES.
C*******
C
      DO 15 I=1,NS
      FRAC(I)=CTS(I)/SUM
      DIV=DIV+FRAC(I)*ALOG(FRAC(I))*CONST
 15   FRAC(I)=FRAC(I)*100.
C
C*******
C*******         WRITE THE CORRECT VALUES.
C*******
C
      DO 20 I=1,NS
 20   WRITE(6,400) (H(K),K=2,9),H(10),(CODE(J,I),J=1,9),
     1CTS(I),FRAC(I),SUM,DIV
 400  FORMAT(8A4,6X,A4,2X,9A1,F8.1,F7.2,F8.1,F6.2)
C
      GO TO 1
C
 99   STOP
      END
```

TABLE 4.5. Program REFORMAT4.

```
C PROGRAM  REFORMAT4  IS RUN TO REORGANIZE LAKE PHYTOPLANKTON RAW DATA
C FILES. THIS PROGRAM IS USED FOR JUNE, AUGUST, SEPTEMBER IN 1976; AND
C ALL MONTHS OF 1977, 1978, 1979, 1980, 1981, 1982.
C UNIT 5 IS ASSIGNED TO INPUT FILE, (RAW DATA FILE).
C UNIT 6 IS ASSIGNED TO OUTPUT FILE, (REFORMATTED DATA FILE).
C UNIT 7 IS ASSIGNED TO ANOTHER OUTPUT FILE FOR THE VALUES OF DL & SW.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      LOGICAL*1 CODE(9,200)
      REAL CTS(200),FRAC(200),H(13)
C
      CONST=-1./ALOG(2.)
C
C*******
C*******        READ THE HEADLINES.
C*******
C
 1    READ(5,100,END=99) (H(I),I=1,13),DL,SW
 100  FORMAT(13A4,1X,2F4.2)
      IDL=DL
      ISW=SW
      WRITE(7,200) IDL,ISW
 200  FORMAT(I2,1X,I3)
C
      CF=(IDL+1.)*41.4516/ISW
      SUM=0.
      NS=1
C
C*******
C*******        READ THE DATA LINES.
C*******
C
 5    READ(5,300,END=10) (CODE(J,NS),J=1,9),CTS(NS)
 300  FORMAT(9A1,F7.0)
      IF(CTS(NS).LT.1.) GO TO 10
      CTS(NS)=CTS(NS)*CF
      SUM=SUM+CTS(NS)
      NS=NS+1
      GO TO 5
C
 10   DIV=0.
      NS=NS-1
C
C*******
C*******        CORRECT THE FRACTION AND DIVERSITY VALUES.
C*******
C
      DO 15 I=1,NS
      FRAC(I)=CTS(I)/SUM
      DIV=DIV+FRAC(I)*ALOG(FRAC(I))*CONST
 15   FRAC(I)=FRAC(I)*100.
C
C*******
C*******        WRITE THE CORRECT VALUES.
C*******
C
      DO 20 I=1,NS
 20   WRITE(6,400) (H(K),K=2,9),H(12),(CODE(J,I),J=1,9),
     1CTS(I),FRAC(I),SUM,DIV
 400  FORMAT(8A4,6X,A4,2X,9A1,F8.1,F7.2,F8.1,F6.2)
C
      GO TO 1
C
 99   STOP
      END
```

TABLE 4.6. Program PLTAXIRCR.

```
RUN *TAXIR
CREATE LAKE.PHYTOPLANKTON,
DAY(FROM 1 TO 31),
MONTH(ORDER,JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC),
YEAR(FROM 70 TO 85),
LOCATION(NAME),
NAME(NAME),
TEMPERATURE(FROM .1 TO 40.0),
CODE(NAME),
GROUP(ORDER,C,D,F,G,H,O,P,R,S),
CELLS(FROM .1 TO 9999.9),
FRAC(FROM .01 TO 100.00),
TOTALCELLS(FROM .1 TO 100000.0),
DIVERSITY(FROM .01 TO 6.00)*
ENTER DATA LOCATION=LREFORM, FORMAT=FIXED,
DAY<5-6>,
MONTH<8-10>,
YEAR<12-13>,
LOCATION<14-25>,
NAME<27-30>,
TEMPERATURE<39-42>,
CODE<45-52>,
GROUP<53>,
CELLS<55-61>,
FRAC<63-68>,
TOTALCELLS<69-76>,
DIVERSITY<79-82>*
SAVE
STOP
```

Added Parameters

When the Lake.Phytoplankton data bank was first created, it contained 12 descriptors but did not include redundancy indexes. It was later felt that there was a need to add such indexes; a 13th descriptor called REDUNDANCY was created, and its values were added to the data bank. Because the values needed to calculate redundancy were not available, they were derived as follows:

Step 1: Use Redundancy program (Table 4.7) to assemble the values corresponding to the numbers of forms, diversities, and total cells from the phytoplankton tables.

Step 2: Use Predundancy program (Table 4.8) to compute redundancy indexes from these values assembled from the phytoplankton tables.

The newly created descriptor REDUNDANCY was then added to the Lake.Phytoplankton data bank by first using the Taxir Statement of Define More Descriptor which is shown as follows:

DMD REDUNDANCY (FROM 0.000 to 1.0000)

Then, Correction Statement was used to enter the values of the descriptor into the data base. An example of this procedure is shown later.

Data Tape and Taxir Table

The complete Lake.Phytoplankton data bank is saved on tape with volume name COOK and ID Code COOK, beginning at first position.

An example of the use of the Taxir program to generate a table for reports is shown in Table 4.9.

TABLE 4.7. Program REDUNDANCY.

```
C PROGRAM REDUNDANCY PROVIDES THE VALUES CORRESPONDING TO THE
C NUMBERS OF FORMS, DIVERSITIES AND TOTAL CELLS .
C UNIT 5 IS INPUT FILE, (PHYTOPLANKTON TABLES).
C UNIT 6 IS ASSIGNED TO OUTPUT FILE.
C
C
C
C
C*******
C*******         INITIALIZE VARIABLES.
C*******
C
      LOGICAL*4 UNDL/'----'/,UNDL1,TITLE/'Tota'/,T1
      LOGICAL*1 DATE(9),CODE(9),EQUC,TOTAL(9)
C
C*******
C*******         READ THE LOOP.
C*******
C
 1    READ(5,100,END=99) UNDL1
 100  FORMAT(3X,A4)
      IF(EQUC(UNDL,UNDL1)) GO TO 2
C
      GO TO 1
C
 2    CALL SKIP(0,2,5)
      READ(5,101) DATE,CODE,N,DIV
 101  FORMAT(2X,9A1,9X,9A1,55X,I4,34X,F6.2)
      N1=(N+1)/2+5
C
      CALL SKIP(0,N1,5)
      READ(5,102) T1,TOTAL
 102  FORMAT(101X,A4,4X,9A1)
      IF(EQUC(T1,TITLE)) GO TO 4
C
      DO 5 I=1,N1
      READ(5,102) T1,TOTAL
      IF(EQUC(T1,TITLE)) GO TO 4
 5    CONTINUE
C
C*******
C*******         CHECK THE ERRORS.
C*******
C
      WRITE(6,104) CODE,DATE
 104  FORMAT('***ERROR*** TOTAL NOT FOUND FOR STATION: ',9A1,1X,
     & 'ON DATE: ',9A1)
      STOP 99
C
C*******
C*******         WRITE THE CORRECT VALUES.
C*******
C
 4    WRITE(6,103) DATE,CODE,N,TOTAL,DIV
 103  FORMAT(1X,9A1,1X,9A1,1X,I4,1X,9A1,1X,F6.2)
C
      GO TO 1
C
 99   STOP
      END
```

TABLE 4.8. Program PREDUNDANCY.

```
C PROGRAM PREDUNDANCY READS IN STATISTICS ON PHYTOPLANKTON CATCHES
C FROM UNIT 5,  COMPUTES A REDUNDANCY INDEX  AND WRITES THE OUTPUT
C TO UNIT 6.
C
C
C
C
C*******
C*******         INITIALIZE VARIABLES.
C*******
C
      LOGICAL*1 DATE(9),CODE(9)
C
C*******
C*******         READ THE STATISTICS.
C*******
C
 3    READ(5,1,END=4) DATE,CODE,N,A,DIV
 1    FORMAT(1X,9A1,1X,9A1,1X,I4,1X,F9.1,1X,F6.2)
C
C*******
C*******         COMPUTE THE REDUNDANCY INDEX.
C*******
C
      B=2*A*3.1416
      B=SQRT(B)
      B=ALOG10(B)
      C=A/2.71828
      C=A*ALOG10(C)
      D=2*3.1416*(A/N)
      D=SQRT(D)
      D=ALOG10(D)
      E=(A/N)/2.71828
      E=(A/N)*ALOG10(E)
      F=(B+C)
      G=(D+E)*N
      DMAX=(3.3219/A)*(F-G)
      T=2*3.1416*(A-(N-1.))
      U=(A-(N-1.))/2.71828
      T=SQRT(T)
      T=ALOG10(T)
      V=(A-(N-1.))
      U=V*ALOG10(U)
      U=U+T
      DMIN=(3.3219/A)*(F-U)
      RI=(DMAX-DIV)/(DMAX-DMIN)
C
C*******
C*******         WRITE THE OUTPUT.
C*******
C
      WRITE(6,2) DATE,CODE,RI
 2    FORMAT('0',9A1,1X,9A1,'REDUNDANCY = ',E10.3)
C
      GO TO 3
C
 4    STOP
      END
```

TABLE 4.9. An example of generating tables using Taxir program.

```
RUN *TAXIR
GET LAKE.PHYTOPLANKTON
Q '+---------------------------------------------------------------+',
'| YEAR | MONTH | LOCATION | CODE | GROUP | CELLS |'<P1>,
'|---------------------------------------------------------------|'<P1>,
('|'<P1,A>, YEAR<P4>, '|'<P9,A>, MONTH<P13>, '|'<P18,A>, LOCATION<P22>,
'|'<P31,A>, CODE<P34>, '|'<P44,A>, GROUP<P49>, '|'<P54,A>, CELLS<P57>,
'|'<P65,A>) FOR ITEMS WITH FRAC>50.0*
```

*No. of items in query response: 167*
*No. of items in data bank: 90078*
*Percentage of response/total data bank: 0.19%*

| YEAR | MONTH | LOCATION | CODE | GROUP | CELLS |
|---|---|---|---|---|---|
| 70 | NOV | DC-2 | OCSPECAA | F | 131.3 |
| | | DC-3 | | | 125.3 |
| | | DC-5 | | | 137.7 |
| | | NDC.21 | | | 157.5 |
| | | NDC.50 | | | 104.6 |
| | | NDC.51 | | | 169.0 |
| | | NDC1-1 | | | 158.0 |
| | | NDC2-0 | CGSPECAA | | 420.0 |
| | | SDC.21 | OCSPECAA | | 61.7 |
| | | SDC.50 | | | 41.9 |
| | | SDC1-0 | | | 178.2 |
| | | SDC2-4 | CGSPECAA | | 155.7 |
| | | SDC4-0 | | | 348.2 |
| | | SDC7-2 | | | 300.3 |
| | | SDC7-3 | | | 327.9 |
| 71 | JUL | DC-2 | GLSPECAA | G | 1118.3 |
| | | DC-4 | | | 1208.7 |
| | | DC-5 | | | 357.3 |
| | | NDC7-1 | | | 2793.7 |
| | | NDC7-2 | | | 1194.8 |
| | | NDC7-3 | | | 1894.6 |
| | SEP | NDC.52 | | | 169.4 |
| | | NDC2-4 | | | 167.0 |
| | | NDC4-0 | MESPECAA | C | 803.7 |
| | | SDC.52 | GLSPECAA | G | 126.7 |
| | | SDC.53 | | | 216.7 |
| | | SDC1-2 | | | 125.7 |
| | | SDC1-3 | | | 187.0 |
| | | SDC4-3 | | | 172.1 |
| | | SDC4-4 | | | 121.1 |
| 72 | JUL | NDC1-2 | TAFENEST | P | 63.1 |
| | | NDC4-3 | FRCROTON | | 165.2 |
| | | NDC7-5 | CGSPECAA | F | 39.9 |
| | | SDC.52 | TAFENEST | P | 104.4 |
| | OCT | DC-6 | CHLIMNET | R | 460.3 |
| | | NDC1-0 | MEGRANUL | C | 834.8 |
| | | NDC1-1 | | | 1400.4 |
| | | NDC2-0 | | | 1105.6 |
| | | NDC2-1 | | | 1500.6 |
| | | NDC4-0 | | | 1254.0 |
| | | NDC4-1 | | | 3964.1 |
| | | NDC4-4 | CHLIMNET | R | 415.3 |
| | | NDC7-1 | MEGRANUL | C | 3984.5 |
| | | NDC7-3 | | | 756.8 |
| | | SDC1-0 | | | 2919.8 |
| | | SDC1-1 | | | 1736.3 |
| | | SDC2-0 | | | 897.8 |
| | | SDC4-1 | | | 1098.8 |
| | | SDC4-4 | CHLIMNET | R | 368.9 |
| 73 | JUL | DC-3 | CYSTELLI | C | 311.4 |
| | | SDC4-3 | | | 827.5 |

An Example for Preparing the Phytoplankton Computer Data Base

In order to add a new data file which we call "RDJUN82" and its redundancy index to the established data base, the following procedure is followed:

```
#SIG XXXX                              (signon MTS)
#XXX                                   (Password)
(The following statements reformat the data set.)
#RUN *FTN SCARDS=REFORMAT4 SPUNCH=FOR.OBJ
#RUN FOR.OBJ 5=RDJUN82 6=REFORMJUN82
(To run the Taxir program to enter the data into the data bank)
#RUN *TAXIR
GET LAKE.PHYTOPLANKTON
ENTER DATA LOCATION=LREFORMJUN82 FORMAT=FIXED,
DAY<5-6>,
   '
   '
   '
   '            Same as in PLTAXIRCR
   '
   '
   '
SAVE
STOP
(To compute the redundancy index)
#RUN *FTN SCARDS=REDUNDANCY SPUNCH=RED.OBJ
#RUN RED.OBJ 5=PHYTOJUN82 6=RJUN82
#RUN *FTN SCARDS=PREDUNDANCY SPUNCH=PRED.OBJ
#RUN PRED.OBJ 5=RJUN82 6=PRJUN82
(Run Taxir program to enter redundancy index into the data bank.)
```

```
#RUN *TAXIR

GET LAKE.PHYTOPLANKTON

CORRECTION (REDUNDANCY=*.***) *MONTH=JUN AND YEAR=82 AND LOCATION=DC-0*
   '
   '
   ' (entering redundancy values for different locations of month of June)
   '
   '
   '

CORRECTION (REDUNDANCY=*.***) *MONTH=JUN AND YEAR=82 AND LOCATION=NDC7-5*

SAVE

STOP

#SIG$ (sign off MTS)
```

*

ENTRAINED PHYTOPLANKTON

The Entrained.Phytoplankton data bank contains 63,748 items and 21 descriptors. The descriptors are listed as follows:

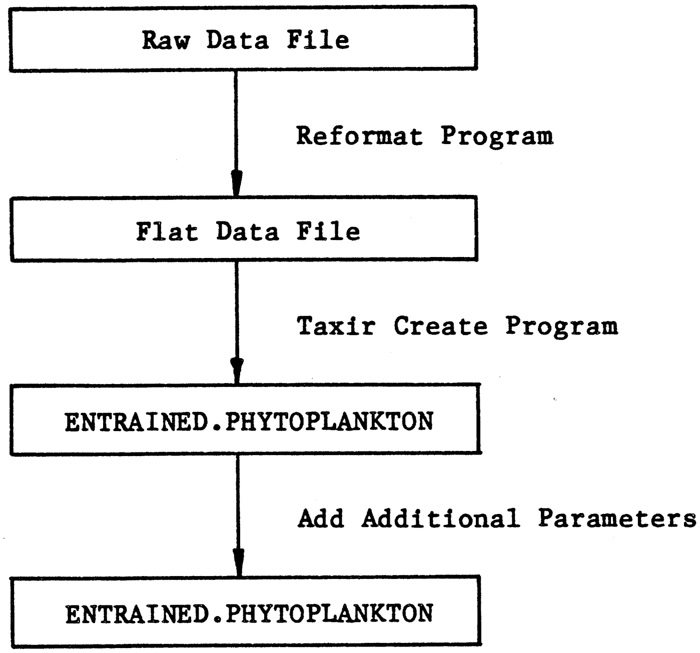
1-DAY
2-MONTH
3-YEAR
4-LOCATION
5-TIME
6-SPECIES CODES
7-MAJOR GROUPS
8-CELLS
9-FRACTION
10-DIVERSITY
11-TEMPERATURE
12-REDUNDANCY
13-CHLOROPHYLL A
14-CHLOROPHYLL B
15-CHLOROPHYLL C
16-PHAEOPHIN
17-CHLOROPHYLL A INCUBATED
18-CHLOROPHYLL B INCUBATED
19-CHLOROPHYLL C INCUBATED
20-PHAEOPHIN
21-HOURS INCUBATED

The number of data items collected for Entrained.Phytoplankton data bank beginning in 1975 is listed in Table 4.10.

TABLE 4.10. The number of data items in Entrained.Phytoplankton data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 75 | FEB | (460) | 78 | JAN | (692) | 80 | JAN | (795) |
| | MAR | (465) | | FEB | (587) | | FEB | (788) |
| | APR | (290) | | MAR | (483) | | MAR | (795) |
| | MAY | (237) | | APR | (661) | | APR | (870) |
| | JUN | (590) | | MAY | (983) | | MAY | (793) |
| | JUL | (619) | | JUN | (1,023) | | JUN | (638) |
| | AUG | (534) | | JUL | (1,254) | | JUL | (487) |
| | SEP | (441) | | AUG | (899) | | AUG | (1,099) |
| | OCT | (659) | | SEP | (1,208) | | SEP | (1,259) |
| | NOV | (604) | | OCT | (1,408) | | OCT | (1,047) |
| | DEC | (559) | | NOV | (871) | | NOV | (634) |
| 76 | JAN | (653) | | DEC | (991) | | DEC | (617) |
| | FEB | (687) | 79 | JAN | (1,074) | 81 | JAN | (879) |
| | MAR | (711) | | FEB | (838) | | FEB | (811) |
| | APR | (673) | | MAR | (1,021) | | MAR | (797) |
| | MAY | (723) | | APR | (537) | | APR | (742) |
| | JUN | (789) | | MAY | (465) | | MAY | (793) |
| | JUL | (1,054) | | JUL | (384) | | JUN | (513) |
| | AUG | (641) | | AUG | (870) | | JUL | (475) |
| | SEP | (1,017) | | SEP | (1,033) | | AUG | (865) |
| | OCT | (706) | | OCT | (1,210) | | SEP | (1,174) |
| | NOV | (686) | | NOV | (678) | | OCT | (632) |
| | DEC | (678) | | DEC | (732) | | NOV | (754) |
| 77 | MAR | (635) | | | | | DEC | (792) |
| | APR | (666) | | | | 82 | JAN | (911) |
| | MAY | (557) | | | | | FEB | (624) |
| | JUN | (769) | | | | | MAR | (888) |
| | JUL | (692) | | | | | APR | (723) |
| | AUG | (563) | | | | | MAY | (706) |
| | SEP | (724) | | | | | | |
| | OCT | (627) | | | | | | |
| | NOV | (559) | | | | | | |
| | DEC | (677) | | | | | | |

The steps for creating the Entrained.Phytoplankton data bank were similar to the ones that were described for the Lake.Phytoplankton data bank. They are shown as follows:




## Original Data and Reformat Program

Original data are stored on the computer tape in a file such as RDENTXXXXX, where the first XXX is a code for month and the second XX is a code for the year (for example, RDENTAPR82). The entrainment data files prior to 1980 are stored on a tape called PHYTO. The 1980, 1981, and 1982 data files can be found on a tape called PHYTO2.

The Reformat Program for entrained phytoplankton is called EREFORMAT. This program is very similar to that used for lake phytoplankton. A listing of this program is given in Table 4.11.

TABLE 4.11. Program EREFORMAT.

```
C PROGRAM EREFORMAT IS USED TO REORGANIZE ENTRAINMENT PHYTOPLANKTON
C RAW DATA FILES.
C UNIT 5 IS ASSIGNED TO INPUT FILE. (RAW DATA FILE).
C UNIT 6 IS ASSIGNED TO OUTPUT FILE. (REFORMATTED DATA FILE).
C UNIT 7 IS ANOTHER OUTPUT FILE FOR THE VALUES OF DL AND  SW.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      LOGICAL*1 CODE(9,200)
      REAL CTS(200),FRAC(200),H(13)
C
      CONST=-1./ALOG(2.)
C
C*******
C*******        READ THE HEADLINES.
C*******
C
 1    READ(5,100,END=99) (H(I),I=1,13),DL,SW
 100  FORMAT(13A4,1X,2F4.2)
      IDL=DL
      ISW=SW
      WRITE(7,200) IDL,ISW
 200  FORMAT(I2,1X,I3)
      CF=(IDL+1.)*41.4516/ISW
C
      SUM=0.
      NS=1
C
C*******
C*******        READ THE DATA LINES.
C*******
C
 5    READ(5,300,END=10) (CODE(J,NS),J=1,9),CTS(NS) ,
 300  FORMAT(9A1,F7.0)
      IF(CTS(NS).LT.1.) GO TO 10
      CTS(NS)=CTS(NS)*CF
      SUM=SUM+CTS(NS)
      NS=NS+1
      GO TO 5
C
 10   DIV=0.
      NS=NS-1
C
C*******
C*******        CORRECT THE FRACTION AND DIVERSITY VALUES.
C*******
C
      DO 15 I=1,NS
      FRAC(I)=CTS(I)/SUM
      DIV=DIV+FRAC(I)*ALOG(FRAC(I))*CONST
 15   FRAC(I)=FRAC(I)*100.
C
C*******
C*******        WRITE THE CORRECT VALUES.
C*******
C
      DO 20 I=1,NS
 20   WRITE(6,400) (H(K),K=2,9),(CODE(J,I),J=1,9),
     1CTS(I),FRAC(I),SUM,DIV,H(12)
 400  FORMAT(8A4,2X,9A1,F8.1,F7.2,F8.1,F6.2,2X,A4)
C
      GO TO 1
C
 99   STOP
      END
```

## Taxir Create Program for Entrained Phytoplankton

To establish the Entrained.Phytoplankton data bank, a Taxir Create program PETAXIRCR (Table 4.12) is used. This program uses the entrained phytoplankton data in flat-file form to store them in a Taxir data bank called Entrained.Phytoplankton.

## Added Parameters

Additional descriptors were added to the Entrained.Phytoplankton data bank. These descriptors are Chlorophyll *a*, Chlorophyll *b*, Chlorophyll *c*, Phaeophin, Chlorophyll *a* Incubated, Chlorophyll *b* Incubated, Chlorophyll *c* Incubated, Phaeophin Incubated, Hours Incubated, and Redundancy Index. The values for Chlorophyll and Phaeophin are obtained by using the computer program ORGANTABLE (Table 4.13). This program selects the parameter information from the chlorophyll and phaeophin tables. Running the above program results in the output data files corresponding to these parameters which are in a form such that Taxir Statement CORRECTION can be applied. In this way, these additional descriptors are stored in the already established Entrained.Phytoplankton data bank.

In order to merge the additional descriptors with appropriate cases of entrained phytoplankton, identification codes (ID codes) of YEAR, MONTH, DAY, LOCATION, and TIME were used. Because the specific times of collection are different at each sampling but the periods basically correspond to morning, noon, and evening periods, these periods were used in place of actual sampling time for ID codes. These periods are shown as follows:

| | |
|---|---|
| Morning | 2-8:30 a.m. |
| Noon | 8:30-14:00 p.m. |
| Evening | 18:00-24:00 p.m. |

TABLE 4.12. Program PETAXIRCR.

```
RUN *TAXIR
CREATE ENTRAINED.PHYTOPLANKTON,
DAY(FROM 1 TO 31),
MONTH(ORDER,JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC),
YEAR(FROM 70 TO 85),
LOCATION(NAME),
TIME(FROM 1 TO 2400),
CODE(NAME),
GROUP(ORDER,C,D,F,G,H,O,P,R,S),
CELLS(FROM .1 TO 9999.9),
FRAC(FROM .01 TO 100.00),
DIVERSITY(FROM .01 TO 6.00),
TEMPERATURE(FROM 0.0 TO 100.0)*
ENTER DATA LOCATION=EREFORM, FORMAT=FIXED,
DAY<7-8>,
MONTH<10-12>,
YEAR<14-15>,
LOCATION<18-20>,
TIME<22-25>,
CODE<35-42>,
GROUP<43>,
CELLS<46-51>,
FRAC<53-58>,
DIVERSITY<69-72>,
TEMPERATURE<74-78>*
SAVE
STOP
```

TABLE 4.13. Program ORGANTABLE.

```
C PROGRAM ORGANTABLE IS FOR REORGANIZING THE CHLOROPHYLL AND PHAEOPHIN
C VALUES FROM THEIR TABLES. THE INPUT FILES ARE:
C UNIT 4; CHLOROPHYLL A TABLE.
C UNIT 5; CHLOROPHYLL B TABLE.
C UNIT 6; CHLOROPHYLL C TABLE.
C UNIT 7; PHAEOPHYTIN TABLE.
C THE OUTPUT FILE  IS  UNIT  8  WHICH  CONTAINS  THE  TAXIR CORRECTION
C STATEMENTS. LATER BY  SETTING  THE SOURCE FILE  EQUAL  TO THE OUTPUT
C FILE THE UNKNOWN  CHLOROPHYLL  AND  PHAEOPHIN VALUES CAN BE ADDED TO
C ENTRAINED.PHYTOPLANKTON DATA BANK.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      REAL LOCTON,MEANA,MEANB,MEANC,MEANP
      INTEGER MONTH,DAY,YEAR,TIME,HRSINC
C
C*******
C*******        READ THE MEAN VALUES, (ONE AT A TIME).
C*******
C
 100  READ(4,1,END=99) MONTH,DAY,YEAR,TIME,LOCTON,HRSINC,MEANA
      READ(5,2) MEANB
      READ(6,2) MEANC
      READ(7,2) MEANP
 1    FORMAT(11X,3(I2,1X),I4,1X,A2,1X,I2,9X,E10.3)
 2    FORMAT(39X,E10.3)
C
C*******
C*******        CHECK THE TOTAL HOURS OF INCUBATION FOR EACH SAMPLE
C*******        AND CONSTRUCT TAXIR CORRECTION STATEMENTS.
C*******
C
      IF(HRSINC.EQ.O) GO TO 10
      IF((TIME.GE.200).AND.(TIME.LE.830)) GO TO 20
      IF((TIME.GT.830).AND.(TIME.LE.1400)) GO TO 30
      WRITE(8,3) MEANA,MEANB,MEANC,MEANP,HRSINC,YEAR,MONTH,DAY,LOCTON
 3    FORMAT(1X,'C (17=',F8.4,')',1X,'(18=',F8.4,')',1X,
     1'(19=',F8.4,')','(20=',F8.4,')',1X,'(21=',I2,')',1X,'*3=',
     1I2,' & 2=',I2,' & 1=',I2,' & 4=',A2,' & 5>1800 & 5<2400*')
      GO TO 100
C
 20   WRITE(8,4) MEANA,MEANB,MEANC,MEANP,HRSINC,YEAR,MONTH,DAY,LOCTON
 4    FORMAT(1X,'C (17=',F8.4,')',1X,'(18=',F8.4,')',1X,
     1'(19=',F8.4,')','(20=',F8.4,')',1X,'(21=',I2,')',1X,'*3=',
     1I2,' & 2=',I2,' & 1=',I2,' & 4=',A2,' & 5>200 & 5<830*')
      GO TO 100
C
 30   WRITE(8,5) MEANA,MEANB,MEANC,MEANP,HRSINC,YEAR,MONTH,DAY,LOCTON
 5    FORMAT(1X,'C (17=',F8.4,')',1X,'(18=',F8.4,')',1X,
     1'(19=',F8.4,')','(20=',F8.4,')',1X,'(21=',I2,')',1X,'*3=',
     1I2,' & 2=',I2,' & 1=',I2,' & 4=',A2,' & 5>830 & 5<1400*')
      GO TO 100
C
 10   IF((TIME.GE.200).AND.(TIME.LE.830)) GO TO 40
      IF((TIME.GT.830).AND.(TIME.LE.1400)) GO TO 50
      WRITE(8,6) MEANA,MEANB,MEANC,MEANP,HRSINC,YEAR,MONTH,DAY,LOCTON
 6    FORMAT(1X,'C (13=',F8.4,')',1X,'(14=',F8.4,')',1X,
     1'(15=',F8.4,')','(16=',F8.4,')',1X,'(21=',I2,')',1X,'*3=',
     1I2,' & 2=',I2,' & 1=',I2,' & 4=',A2,' & 5>1800 & 5<2400*')
      GO TO 100
C
```

Continued on next page.

TABLE 4.13. (Continued).

```
 40    WRITE(8,7) MEANA,MEANB,MEANC,MEANP,HRSINC,YEAR,MONTH,DAY,LOCTON
 7     FORMAT(1X,'C (13=',F8.4,')',1X,'(14=',F8.4,')',1X,
     1'(15=',F8.4,')','(16=',F8.4,')',1X,'(21=',I2,')',1X,'"3=',
     1I2,' & 2=',I2,' & 1=',I2,' & 4=',A2,' & 5>200 & 5<830"')
       GO TO 100
C
 50    WRITE(8,8) MEANA,MEANB,MEANC,MEANP,HRSINC,YEAR,MONTH,DAY,LOCTON
 8     FORMAT(1X,'C (13=',F8.4,')',1X,'(14=',F8.4,')',1X,
     1'(15=',F8.4,')','(16=',F8.4,')',1X,'(21=',I2,')',1X,'"3=',
     1I2,' & 2=',I2,' & 1=',I2,' & 4=',A2,' & 5>830 & 5<1400"')
       GO TO 100
C
 99    STOP
       END
```

Computer commands used in this process are shown below:

```
#RUN *FIN SCARDS=ORGANTABLE SPUNCH=-LOAD

#RUN -LOAD 4=CHLORAXX

           5=CHLORBXX

           6=CHLORCXX          (XX=YEAR)

           7=PHAXX             (4,5,6,7)=INPUT FILES

           8=CHLPHAXX          8=OUTPUT FILE

#SOURCE CHLPHAXX
```

Data Tape

The complete Entrained.Phytoplankton data bank with 21 descriptors is saved on the tape called COOK beginning at the 2nd position.

LAKE ZOOPLANKTON

The Lake.Zooplankton data bank contains 32,113 items and 10 descriptors. The descriptors are:

1-STATION

2-MONTH

3-YEAR

4-DEPTH

5-TAXON #

6-COUNT

7-PCCOMP

8-TAXON

9-TEMPERATURE

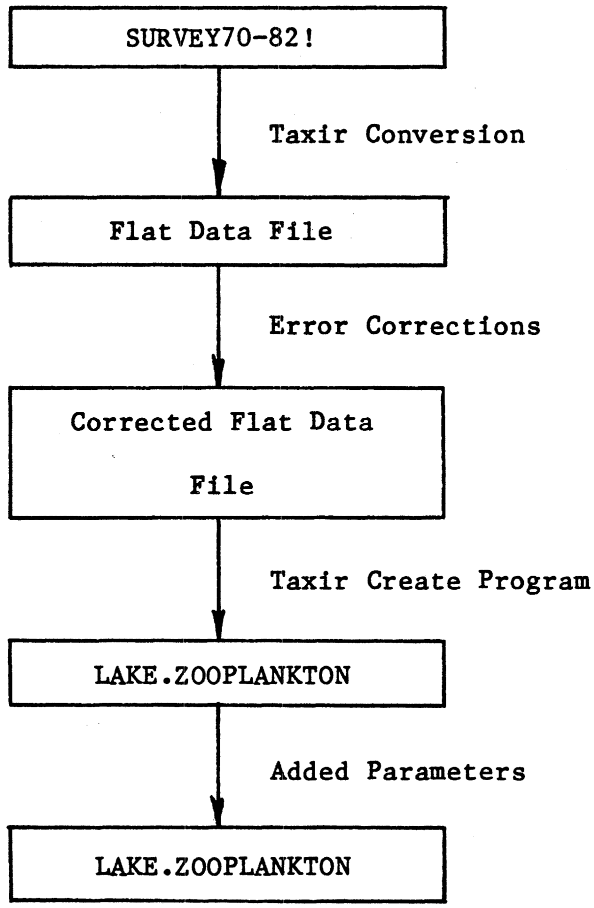
10-SECCHI DISC

The monthly number of data items collected for lake zooplankton beginning in July 1970 is listed in Table 4.14.

TABLE 4.14. The number of data items in Lake.Zooplankton data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 70 | JUL | ( 459) | 75 | APR | ( 441) | 79 | APR | ( 431) |
| | SEP | ( 508) | | MAY | ( 228) | | MAY | ( 211) |
| | NOV | ( 461) | | JUN | ( 312) | | JUN | ( 204) |
| 71 | APR | ( 189) | | JUL | ( 630) | | JUL | ( 582) |
| | JUL | ( 464) | | AUG | ( 718) | | AUG | ( 298) |
| | SEP | ( 603) | | SEP | ( 322) | | SEP | ( 280) |
| | NOV | ( 402) | | OCT | ( 611) | | OCT | ( 677) |
| 72 | APR | ( 421) | | DEC | ( 281) | | NOV | ( 296) |
| | MAY | ( 90) | 76 | APR | ( 391) | 80 | APR | ( 488) |
| | JUN | ( 104) | | MAY | ( 488) | | MAY | ( 253) |
| | JUL | ( 353) | | JUN | ( 260) | | JUN | ( 262) |
| | AUG | ( 121) | | JUL | ( 637) | | JUL | ( 489) |
| | SEP | ( 116) | | AUG | ( 336) | | AUG | ( 299) |
| | OCT | ( 393) | | SEP | ( 323) | | SEP | ( 342) |
| | NOV | ( 115) | | OCT | ( 593) | | OCT | ( 625) |
| 73 | APR | ( 417) | 77 | APR | ( 442) | | NOV | ( 251) |
| | MAY | ( 134) | | MAY | ( 215) | 81 | APR | ( 401) |
| | JUN | ( 167) | | JUN | ( 250) | | MAY | ( 243) |
| | JUL | ( 566) | | JUL | ( 635) | | JUN | ( 214) |
| | AUG | ( 173) | | AUG | ( 318) | | JUL | ( 456) |
| | SEP | ( 171) | | SEP | ( 321) | | AUG | ( 305) |
| | OCT | ( 582) | | OCT | ( 660) | | SEP | ( 312) |
| 74 | APR | ( 389) | | NOV | ( 288) | | OCT | ( 547) |
| | MAY | ( 223) | | DEC | ( 238) | | NOV | ( 290) |
| | JUN | ( 247) | 78 | APR | ( 454) | 82 | APR | ( 416) |
| | JUL | ( 511) | | MAY | ( 260) | | MAY | ( 186) |
| | AUG | ( 307) | | JUN | ( 282) | | | |
| | SEP | ( 307) | | JUL | ( 570) | | | |
| | OCT | ( 606) | | AUG | ( 304) | | | |
| | | | | SEP | ( 337) | | | |
| | | | | OCT | ( 702) | | | |
| | | | | NOV | ( 309) | | | |

The steps for constructing this data bank follow the flow diagram shown below:




## Lake.Zooplankton Data Bank

The Lake.Zooplankton data bank was established using the steps listed above. These steps are the same as those taken for the establishment of the phytoplankton data bank, except that the length for each descriptor name was reduced to six characters, which is the maximum number of labeling characters that MIDAS permits. Two descriptors, TEMPERATURE and SECCHI DISC, were also added to this data set. These additions were made using the Taxir Statement CORRECTION. Both the steps for adding additional parameters and procedures which were used to create the Lake.Zooplankton data bank are shown below:

1) Convert the SURVEY70-82! to a line file.

#RUN *TAXIR

GET SURVEY70-82!

Q<DATA, SURVEY> ALL *ALL*

STOP

2) Correct the errors in the SURVEY data file.

#Ed SURVEY

:A@A /F ;JANUARY;JANbbbb;

: ' '
' '
' '
' '
' '

:A@A /F ;DECEMBER;DECbbbbb;

:A@A /F ;DC-0-0;DC-0bb;

: ' '
' '
' '
' '
' '

:A@A /F ;NDC-7-5;NDC 7-5b;

:STOP

#SOURCE ZLTAXIRCR

3) (Add values of SECCHI DISC and TEMPERATURE to the data bank.

CORRECTION (SECCHIDISC=....) AND (TEMPERATURE=....)

```
*MONTH=.... AND YEAR=.... AND STATION=....*
      '
      '
      '
      '
      '
SAVE
STOP
```

ZLTAXIRCR is presented in Table 4.15.

## Data Tape

The final version of the Lake.Zooplankton data bank is saved on the tape called COOK at the 3rd position.

TABLE 4.15. Program ZLTAXIRCR.

```
RUN *TAXIR
CREATE LAKE.ZOOPLANKTON,
STATION(NAME),
MONTH(ORDER,JAN,FEB,MAR,APR,MAY,JUN,JUL,AUG,SEP,OCT,NOV,DEC),
YEAR(FROM 69 TO 85),
DEPTH(FROM 0 TO 60),
TAXON #(FROM 10001 TO 99999),
COUNT(FROM 0 TO 1000000),
PCCOMP(FROM 0.00 TO 100.00),
TAXON(NAME),
TEMPERATURE(FROM 0.1 TO 40.0),
SECCHI DISC(FROM 0.00 TO 100.00)*
ENTER DATA LOCATION=SURVEY, FORMAT=FIXED,
STATION<1-8>,
MONTH<10-18>,
YEAR<20-21>,
DEPTH<23-24>,
TAXON #<26-30>,
COUNT<32-38>,
PCCOMP<40-45>,
TAXON<47-81>,
TEMPERATURE<83-86>,
SECCHI DISC<88-93>*
SAVE
STOP
```

ENTRAINED ZOOPLANKTON

The Entrained.Zooplankton data bank consists of 19,703 items and 11 descriptors. The descriptors are listed below:

1-TYPE

2-GRATE

3-MONTH

4-DAY

5-YEAR

6-TIME

7-TAXON #

8-COUNT

9-PCCOMP

10-TAXON

11-TEMPERATURE

The monthly number of data items collected for entrained zooplankton beginning in 1975 is listed in Table 4.16.

TABLE 4.16. The number of data items in Entrained.Zooplankton data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 75 | JAN | ( 30) | 78 | JAN | (165) | 80 | JAN | (304) |
| | FEB | (149) | | FEB | ( 95) | | FEB | (170) |
| | MAR | (118) | | MAR | ( 88) | | MAR | (151) |
| | APR | (126) | | APR | (120) | | APR | (185) |
| | MAY | (121) | | MAY | (184) | | MAY | (224) |
| | JUN | (166) | | JUN | (336) | | JUN | (163) |
| | JUL | (204) | | JUL | (523) | | JUL | (146) |
| | AUG | (218) | | AUG | (550) | | AUG | (266) |
| | SEP | (222) | | SEP | (355) | | SEP | (330) |
| | OCT | (203) | | OCT | (445) | | OCT | (276) |
| | NOV | (182) | | NOV | (197) | | NOV | (169) |
| | DEC | (142) | | DEC | (401) | | DEC | (155) |
| 76 | JAN | (153) | 79 | JAN | (235) | 81 | JAN | (203) |
| | FEB | (101) | | FEB | (179) | | FEB | (134) |
| | MAR | (101) | | MAR | (210) | | MAR | (150) |
| | APR | (109) | | APR | (151) | | APR | (119) |
| | MAY | (156) | | MAY | (191) | | MAY | (216) |
| | JUN | (258) | | JUN | (166) | | JUN | (117) |
| | JUL | (312) | | JUL | (396) | | JUL | (152) |
| | AUG | (509) | | AUG | (535) | | AUG | (326) |
| | SEP | (192) | | SEP | (436) | | SEP | (417) |
| | OCT | (189) | | OCT | (383) | | OCT | (138) |
| | NOV | (184) | | NOV | (273) | | NOV | (158) |
| | DEC | (173) | | DEC | (266) | | DEC | (258) |
| 77 | FEB | ( 33) | | | | 82 | JAN | (220) |
| | MAR | (187) | | | | | FEB | (128) |
| | APR | (160) | | | | | MAR | (213) |
| | MAY | (138) | | | | | APR | (179) |
| | JUN | (175) | | | | | MAY | (171) |
| | JUL | (436) | | | | | | |
| | AUG | (598) | | | | | | |
| | SEP | (312) | | | | | | |
| | OCT | (290) | | | | | | |
| | NOV | (174) | | | | | | |
| | DEC | (164) | | | | | | |

Entrained.Zooplankton Data Bank

The Entrained.Zooplankton data bank contains 11 descriptors; the last of which, TEMPERATURE, was added to this data base after it was created by the Taxir program. The final version of the Entrained.Zooplankton data bank is stored on the tape COOK at the 4th position.

LAKE BENTHOS

The Lake.Benthos data bank has 72,504 items and 16 descriptors. The descriptors are:

1-YEAR

2-MONTH

3-REGION

4-ZONE

5-DEPTH

6-CON.FACT

7-AREA

8-STATION

9-GROUP

10-CODE

11-CELLS

12-PRIM SED

13-SEC SED

14-TERT SED

15-QUAT SED

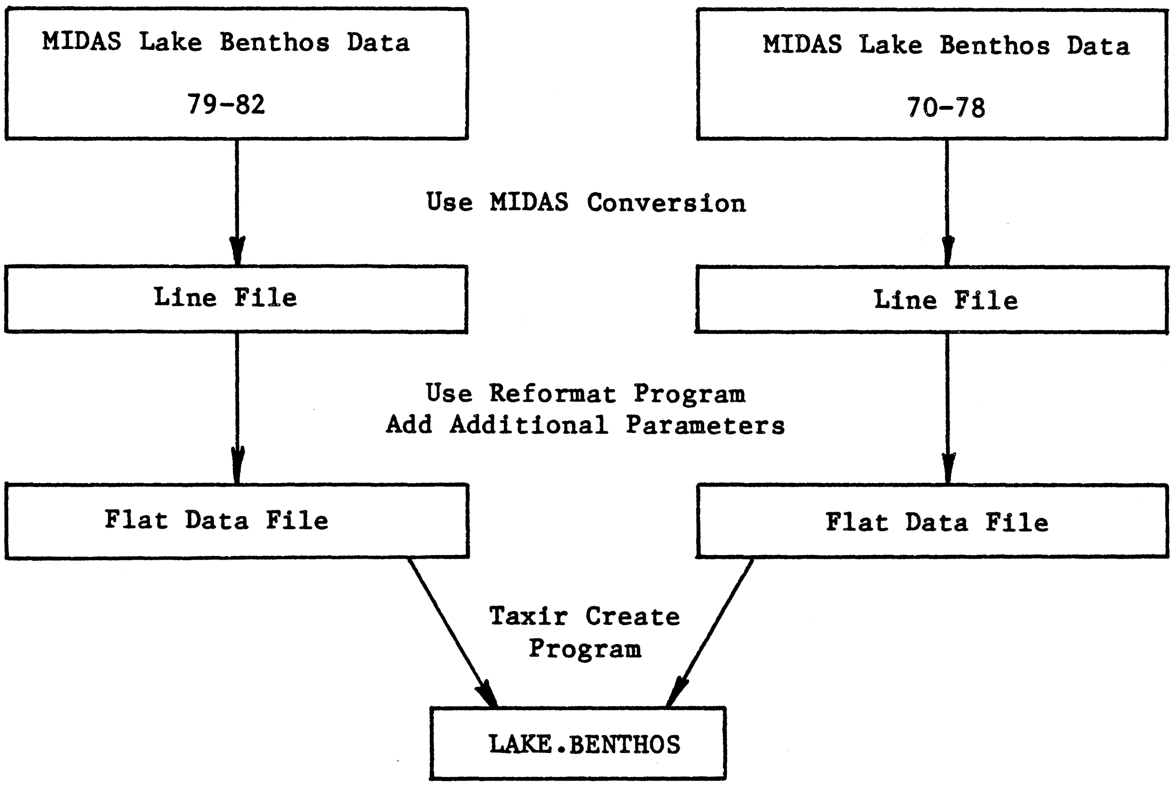
16-VOLUME

The number of data items collected for the Lake.Benthos data bank is shown by months in Table 4.17.

TABLE 4.17. The number of data items in Lake.Benthos data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 70 | 7 | ( 595) | 75 | 4 | (1,065) | 78 | 4 | ( 980) |
| | 11 | ( 817) | | 5 | ( 696) | | 5 | ( 482) |
| 71 | 4 | ( 702) | | 6 | ( 784) | | 6 | ( 564) |
| | 7 | ( 857) | | 7 | (1,862) | | 7 | (1,881) |
| | 11 | ( 610) | | 8 | (1,030) | | 8 | ( 585) |
| 72 | 4 | ( 898) | | 9 | ( 934) | | 9 | ( 652) |
| | 5 | ( 318) | | 10 | (1,301) | | 10 | (1,572) |
| | 6 | ( 363) | | 12 | ( 561) | | 11 | ( 599) |
| | 7 | (1,970) | 76 | 4 | (1,081) | 79 | 4 | ( 698) |
| | 8 | ( 371) | | 5 | ( 667) | | 5 | ( 556) |
| | 9 | ( 294) | | 6 | ( 434) | | 6 | ( 627) |
| | 10 | (1,996) | | 7 | (1,360) | | 7 | ( 949) |
| | 11 | ( 335) | | 8 | ( 536) | | 8 | ( 576) |
| 73 | 4 | (1,774) | | 9 | ( 495) | | 9 | ( 531) |
| | 5 | ( 222) | | 10 | (1,036) | | 10 | ( 911) |
| | 6 | ( 440) | 77 | 4 | (1,153) | | 11 | ( 545) |
| | 7 | (3,074) | | 5 | ( 533) | 80 | 4 | ( 625) |
| | 8 | ( 448) | | 6 | ( 515) | | 5 | ( 570) |
| | 9 | ( 486) | | 7 | (1,884) | | 6 | ( 539) |
| | 10 | (2,173) | | 8 | ( 565) | | 7 | ( 922) |
| 74 | 4 | (2,030) | | 9 | ( 614) | | 8 | ( 582) |
| | 5 | ( 600) | | 10 | (1,373) | | 9 | ( 625) |
| | 6 | ( 753) | | 11 | ( 642) | | 10 | ( 882) |
| | 7 | (1,629) | | 12 | ( 543) | | 11 | ( 445) |
| | 8 | ( 879) | | | | 81 | 4 | ( 701) |
| | 9 | ( 844) | | | | | 5 | ( 478) |
| | 10 | (1,327) | | | | | 6 | ( 532) |
| | | | | | | | 7 | ( 839) |
| | | | | | | | 8 | ( 586) |
| | | | | | | | 9 | ( 598) |
| | | | | | | | 10 | ( 881) |
| | | | | | | | 11 | ( 490) |
| | | | | | | 82 | 4 | ( 556) |
| | | | | | | | 5 | ( 476) |

Procedures for establishing the Lake.Benthos data bank are shown by the following flow diagram:




Lake.Benthos Data Bank

The original Lake Benthos data were stored in MIDAS internal files COOKMERGE and COOK79-82. The former houses the data between 1970 and 1978, and the latter contains the data between 1979 and 1982. The procedures for establishing this data base are: (1) to change these MIDAS internal files to line files, then (2) to change the line files to flat files, and (3) to use a Taxir Create program to create the Lake.Benthos data bank.

To accomplish the first step, we used the following computer statements to convert COOKMERGE to a line file:

```
(while in MTS mode)
#RUN STAT:MIDAS
?READ INTERNAL FILE=COOKMERGE VARIABLE=ALL
(writing the variables of interest in the required order in the file
LINVMIDAS, an MTS line file)
?WRITE VARIABLE=1-5,9,155-156,10-147,150-154,200-205,300-303,400-406,500-510
FILE=LINVMIDAS CASES=ALL FORMAT=(2(F3.0,2X),2(F2.0,2X),F4.1,2X,F5.2,2X,F2.0,2X,
F4.0,2X,166(F10.2,2X),4(F2.0,2X),F4.1)
?FINISH
```

Note that the order of the variables in the file LINVMIDAS follows the sequence of variables, YEAR, MONTH, REGION, ZONE, DEPTH, CON.FACT, AREA, STATION, 166 SPECIES VALUES, PRIMSED, SECSED, TERTSED, QUATSED, and VOLUME.

With a slight change, the above statements were applied to change COOK7982 to a line file. The complete statements are as follows:

```
#RUN STAT:MIDAS
?READ INTERNAL FILE=COOK7982 VARIABLE=ALL
?WRITE VARIABLE=1-31,34-61 FILE=MCOOK7982 CASES=ALL
FORMAT=(2(F3.0,2X),F2.0,2X,2(F2.0,2X),F4.1,2X,F5.2,2X,4(F2.0,2X),F4.1,2X,46(F10.
2,2X),F4.0)
?FINISH
```

The order of the variables in file MCOOK7982 is different from that in the LINVMIDAS and is shown as follows: YEAR, MONTH, AREA, REGION, ZONE, DEPTH, CON.FACT, PRIMSED, SECSED, TERTSED, QUATSED, VOLUME, 46 SPECIES VALUES, and STATION.

## Reformat Programs

Each benthos species in the BLINVMIDAS is treated as a parameter. There are 166 species included in this file, but only 46 species occur frequently. A large amount of space in the file is, therefore, occupied by species occurring only rarely; this represents an inefficient utilization of space. We decided to change the form of data storage so as to make more efficient use of file space. Programs BLARRAN1 and BLARRAN2 were written for this purpose. These programs ignore those species with counts of 0.0; the remaining species are placed in a flat-file form with one species per line. The program BLARRAN1 is used for restructuring the 1970-1978 data while BLARRAN2 is used for rearranging the 1978-1982 data. The results of these programs are stored in files REMCOOK7982 and BLREMIDAS. The command statements for using these programs are listed below:

```
#RUN *FTN SCARDS=BLARRAN1 SPUNCH=OBJ1
#RUN *FTN SCARDS=BLARRAN2 SPUNCH=OBJ2
#RUN OBJ1 5=BLINVMIDAS 6=BLREMIDAS
#RUN OBJ2 5=MCOOK7982 6=REMCOOK7982
```

The contents of these programs are listed in Tables 4.18 and 4.19.

TABLE 4.18. Program BLARRAN1.

```
C PROGRAM BLARRAN1 IS USED TO ORGANIZE LAKE BENTHOS DATA SETS FOR
C THE PERIOD OF 1970 TO 1978.  THE  OUTPUT FILE IS IN THE FORM OF
C FLAT FILE  WHICH  LATER CAN BE ADDED TO LAKE.BENTHOS DATA BANK.
C UNIT 5 IS THE INPUT FILE.
C UNIT 6 IS ASSIGNED TO OUTPUT FILE. (FLAT FILE).
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      REAL YEAR,MONTH,REGIN,ZONE,DEPTH,CONFAC,AREA,STAT,PRSD,
     1SESD,TESD,QUSD,VOLUME,SPCONT(166)
      REAL*8 SPNAME(166)
      INTEGER    IYEAR,IMONTH,IREGIN,IZONE,IAREA,ISTAT,IPRSD,
     1ISESD,ITESD,IQUSD
      LOGICAL*1 GROUPN(166)
C
C*******
C*******        INITIALIZE DATA.
C*******
C
      DATA SPNAME/8H  C.FLUV,8H C.ANTHR,8HCHIRNMUS,8H   CLADO,
     18H C.ROLLI,8HCRYPTO 1,8HCRYPTO 2,8HCRYPTO 3,8H  CRICOT,
     18HDEMICRYP,8HH.CHANGI,8H  H.OLIV,8H  HYDROB,8H  MICROP,
     18H   M.TUB,8HP.ABORT,8HP.NEREIS,8HP.UNDINE,8HP.CAMPTO,
     18H  P.WINN,8H P.FALLX,8H  P.SCAL,8H  POLY 2,8H  P.LONG,
     18H PROCLAD,8H P.SIMUL,8HRHEOTANY,8H  R.DEMJ,8H S.TYLUS,
     18H TANYTAR,8HTHIEN GR,8H O. CHIR,8H  A.LEYD,8HA.LOMOND,
     18H C.DIAPH,8H  C.DIAS,8HC.SETOSU,8H    DERO,8H  N.PARD,
     18HN.PSEUDO,8H   N.VAR,8H  N.SIMP,8H   P.LIT,8HP.SIMPLX,
     18H  P.MICH,8HP.FORELI,8HP.LONGIS,8HP.OSBORN,8HS.APPEND,
     18H S.JOSIN,8H S.LACUS,8H U.UNCIN,8H V.INTER,8HIM WO HC,
     18H IM W HC,8HL.ANGUST,8H  L.CLAP,8HL.CERVIX,8H  L.HOFF,
     18H  L.PROF,8H  L.SPIR,8H  L.UDEK,8H P.FREYI,8H PELO MM,
     18H PELO ML,8H P.SUPER,8HP.BEDOTI,8H  P.MOLD,8H   P.VEJ,
     18H  A.AMER,8H A.LIMNO,8H  A.PLUR,8H  I.TEMP,8HT.TUBIFX,
     18H  R.COCC,8H   V.SIN,8H   V.TRI,8HAMNICOLA,8HBYTHINIA,
     18H  SOMATO,8H LYMNAEA,8H   PHYSA,8HO. GASTR,8H S.NITID,
     18H S.STRIA,8H S.TRANS,8H  S.CORN,8HP.ADAMSI,8H P.CASER,
     18H  P.COMP,8H  P.CONV,8H  P.FALL,8H  P.FERR,8H  P.HENS,
     18H P.IDAHO,8H P.LILLJ,8H  P.MILL,8H   P.NIT,8H  P.PAUP,
     18H P.SUPIN,8H   P.SUB,8H   P.VAR,8H  P.WALK,8H  H.STAG,
     18H D.PARVA,8H  N.OBSC,8H  G.COMP,8HO. HIRUD,8HO INSECT,
     18HP.HOYI 1,8HP.HOYI 2,8HP.HOYI 3,8HP.HOYI 4,8HP.HOYI G,
     18HP.HOYI S,8HP.HOYI M,8H  TPONTO,8H  TMYSIS,8H   TGAMM,
     18H  THYALL,8H   TCHIR,8H   TNAID,8H  TTUBIF,8H  TENCHY,
     18H  TSTYLO,8H  TOLIGO,8H TGASTRO,8H TSPHAER,8H  TPISIP,
     18H TPELECY,8H THYDRAC,8H  THIRUD,8H  THYDRA,8HTTURBELL,
     18H  TOTHER,8H TANIMAL,8H TAEOLOS,8H ASELLUS,8H O. NAID,
     18H  O.SERP,8H  N.COMM,8H P.FRICI,8HN.BRETSC,8HN.BEHNGI,
     18HO. TUBIF,8H P.VARIG,8H  P.HAMM,8H   A.PIG,8HM. PISID,
     18HU. PISID,8H S.RHOMB,8H P.AMNIC,8H S.SECUR,8HO. SPHAE,
     18HP.VENTRI,8H M. CHIR,8H   KIEFF,8H  C.HALO,8HPARACLAD,
     18HHARNISCH,8H PHAENOP,8H   DICRO,8H  GLYPTO,8HPARATANY,
     18HRHEOTANY,8H  STICTO/
C
      DATA    GROUPN/32*1HC,21*1HN,22*1HT,8*1HG,4*1HS,16*1HP,
     15*1HH,1HO,8*1HA,1HO,2*1HA,1HC,1HN,1HT,3*1HO,1HG,1HS,1HP,
     12*1HO,1HH,6*1HO,6*1HN,4*1HT,2*1HP,1HS,1HP,2*1HS,1HP,11*1HC/
C
C*******
C*******        READ DATA LINES CONTAINING 166 SPECIES COUNTS.
C*******
C
```

Continued on next page.

TABLE 4.18. (Continued).

```
 100  READ(5,10,END=99) YEAR,MONTH,REGIN,ZONE,DEPTH,CONFAC,AREA,
     1STAT,(SPCONT(J),J=1,166),PRSD,SESD,TESD,QUSD,VOLUME
 10   FORMAT(2(F3.0,2X),2(F2.0,2X),F4.1,2X,F5.2,2X,F2.0,2X,F4.0,
     12X,166(F10.2,2X),4(F2.0,2X),F4.1)
C
      IYEAR=YEAR
      IMONTH=MONTH
      IREGIN=REGIN
      IZONE=ZONE
      IAREA=AREA
      ISTAT=STAT
      IPRSD=PRSD
      ISESD=SESD
      ITESD=TESD
      IQUSD=QUSD
C
C*******
C*******        WRITE THE FLAT FILE WITH THE SPECIFICATIONS
C*******        OF ONE SPECIES AT EACH LINE.
C*******
C
      DO 20 I=1,166
      IF(SPCONT(I).LE.O.) GO TO 20
      WRITE(6,30) IYEAR,IMONTH,IREGIN,IZONE,DEPTH,CONFAC,IAREA,
     1ISTAT,GROUPN(I),SPNAME(I),SPCONT(I),IPRSD,ISESD,ITESD,
     1IQUSD,VOLUME
 30   FORMAT(2(I3,3X),2(I2,3X),F4.1,3X,F5.2,3X,I2,3X,I4,3X,A1,
     13X,A8,3X,F10.2,3X,4(I2,3X),F4.1)
 20   CONTINUE
C
      GO TO 100
C
 99   STOP
      END
```

TABLE 4.19. Program BLARRAN2.

```
C PROGRAM BLARRAN2 IS USED TO ORGANIZE LAKE BENTHOS DATA SETS FOR
C THE PERIOD OF 1979 TO 1982.  THE  OUTPUT FILE IS IN THE FORM OF
C FLAT FILE  WHICH  LATER CAN BE ADDED TO LAKE.BENTHOS DATA BANK.
C UNIT 5 IS THE INPUT FILE.
C UNIT 6 IS ASSIGNED TO OUTPUT FILE. (FLAT FILE).
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      REAL YEAR,MONTH,AREA,REGIN,ZONE,DEPTH,CONFAC,PRSD,SESD,
     1TESD,QUSD,VOLUME,SPCONT(46),STAT
      REAL*8 SPNAME(46)
      INTEGER    IYEAR,IMONTH,IAREA,IREGIN,IZONE,IPRSD,ISESD,
     1ITESD,IQUSD,ISTAT
      LOGICAL*1 GROUPN(46)
C
C*******
C*******        INITIALIZE DATA.
C*******
C
      DATA SPNAME/8HP.HOYI 1,8HP.HOYI 2,8HP.HOYI 3,8HP.HOYI 4,
     18HP.HOYI G,8HP.HOYI S,8HP.HOYI M,8H  TPONTO,8H  TMYSIS,
     18H   TGAMM,8H  THYALL,8H ASELLUS,8HV.LEWISI,8H   V.SIN,
     18HAMNICOLA,8HBYTHINIA,8H  SOMATO,8H LYMNAEA,8H   PHYSA,
     18H TGASTRO,8H  TPISID,8HS.MARGIN,8H S.NITID,8HS.SIMILE,
     18H S.STRIA,8H S.TRANS,8H TSPHAER,8H TPELECY,8H   TCHIR,
     18H  TSTYLO,8H   TNAID,8H  TTUBIF,8H  TENCHY,8H  TOLIGO,
     18HGLOSSOPH,8H  H.STAG,8H D.PARVA,8H  N.OBSC,8HO. HIRUD,
     18H  THIRUD,8H THYDRAC,8H  THYDRA,8HTTURBELL,8H  TOTHER,
     18HO INSECT,8H TANIMAL/
C
      DATA    GROUPN/8*1HA,1HO,2*1HA,1HO,8*1HG,1HP,6*1HS,1HO,
     11HC,1HO,1HN,1HT,2*1HO,6*1HH,6*1HO/
C
C*******
C*******        READ DATA LINES CONTAINING 46 SPECIES COUNTS.
C*******
C
 100  READ(5,10,END=99) YEAR,MONTH,AREA,REGIN,ZONE,DEPTH,
     1CONFAC,PRSD,SESD,TESD,QUSD,VOLUME,(SPCONT(J),J=1,46),STAT
 10   FORMAT(2(F3.0,2X),F2.0,2X,2(F2.0,2X),F4.1,2X,
     1F5.2,2X,4(F2.0,2X),F4.1,2X,46(F10.2,2X),F4.0)
C
      IYEAR=YEAR
      IMONTH=MONTH
      IAREA=AREA
      IREGIN=REGIN
      IZONE=ZONE
      IPRSD=PRSD
      ISESD=SESD
      ITESD=TESD
      IQUSD=QUSD
      ISTAT=STAT
C
      IF(IYEAR.EQ.10) IYEAR=79
      IF(IYEAR.EQ.11) IYEAR=80
      IF(IYEAR.EQ.12) IYEAR=81
      IF(IYEAR.EQ.13) IYEAR=82
C
      IF(CONFAC.EQ.1.) CONFAC=60.60
      IF(CONFAC.EQ.2.) CONFAC=20.40
      IF(CONFAC.EQ.3.) CONFAC=30.30
```

**Continued on next page.**

TABLE 4.19. (Continued).

```
C
C*******
C*******        WRITE THE FLAT FILE WITH THE SPECIFICATIONS
C*******        OF ONE SPECIES AT EACH LINE.
C*******
C
      DO 20 I=1,46
      IF(SPCONT(I).LE.0.) GO TO 20
      WRITE(6,30) IYEAR,IMONTH,IREGIN,IZONE,DEPTH,CONFAC,
     1IAREA,ISTAT,GROUPN(I),SPNAME(I),SPCONT(I),IPRSD,
     1ISESD,ITESD,IQUSD,VOLUME
 30   FORMAT(2(I3,3X),2(I2,3X),F4.1,3X,F5.2,3X,I2,3X,I4,
     13X,A1,3X,A8,3X,F10.2,3X,4(I2,3X),F4.1)
 20   CONTINUE
C
      GO TO 100
C
 99   STOP
      END
```

Taxir Create Program

Program BLTAXIRCR is a Taxir Create program for establishing the Lake.Benthos data bank. This Create program uses the result of the reformat programs establishing a Taxir data base, LAKE.BENTHOS, which is saved on the COOK tape at position 5. The Taxir Create program for the Lake.Benthos data bank is shown in Table 4.20.

TABLE 4.20. Program BLTAXIRCR.

```
RUN *TAXIR
CREATE LAKE.BENTHOS,
YEAR(FROM 70 TO 85),
MONTH(FROM -0 TO 12),
REGION(FROM -0 TO 9),
ZONE(FROM -0 TO 9),
DEPTH(FROM -0.0 TO 99.9),
CON.FACT(FROM -0.00 TO 99.99),
AREA(FROM -0 TO 9),
STATION(FROM -0 TO 999),
GROUP(ORDER,A,C,G,H,N,O,P,S,T),
CODE(NAME),
CELLS(FROM 0.01 TO 1000000.99),
PRIM SED(FROM -0 TO 9),
SEC SED(FROM -0 TO 9),
TERT SED(FROM -0 TO 9),
QUAT SED(FROM -0 TO 9),
VOLUME(FROM -0.0 TO 99.9)*
ENTER DATA LOCATION=BLREMIDAS, FORMAT=FIXED,
YEAR<1-3>,
MONTH<7-9>,
REGION<13-14>,
ZONE<18-19>,
DEPTH<23-26>,
CON.FACT<30-34>,
AREA<38-39>,
STATION<43-46>,
GROUP<50>,
CODE<54-61>,
CELLS<65-74>,
PRIM SED<78-79>,
SEC SED<83-84>,
TERT SED<88-89>,
QUAT SED<93-94>,
VOLUME<98-101>*
SAVE
STOP
```

ENTRAINED BENTHOS

The Entrained.Benthos data bank contains 11 descriptors and 7,911 items. The descriptors are:

1-SAMPLE

2-YEAR

3-MONTH

4-WEEK

5-PERIOD

6-PUMP

7-REPLICATION

8-CUBIC.METER

9-GROUP

10-CODE

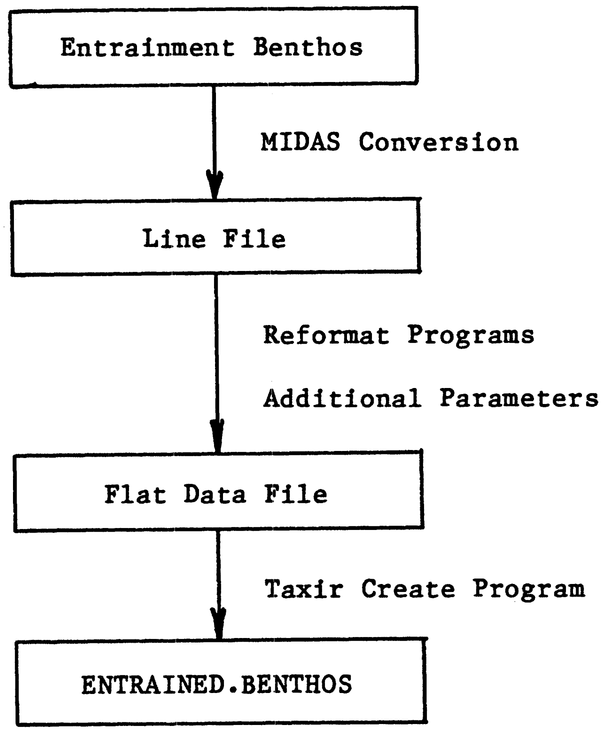
11-CELLS

The monthly number of data items stored in the Entrained.Benthos data bank is shown in Table 4.21.

TABLE 4.21. The number of data items in Entrained.Benthos data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 75 | 1 | ( 54) | 78 | 1 | ( 75) | 81 | 1 | ( 62) |
| | 2 | ( 94) | | 2 | ( 22) | | 2 | ( 42) |
| | 3 | ( 25) | | 3 | ( 21) | | 3 | ( 79) |
| | 4 | ( 48) | | 4 | ( 46) | | 4 | ( 59) |
| | 5 | ( 47) | | 5 | ( 98) | | 5 | (131) |
| | 6 | ( 65) | | 6 | (195) | | 6 | (281) |
| | 7 | ( 69) | | 7 | (159) | | 7 | (185) |
| | 8 | (109) | | 8 | (189) | | 8 | (200) |
| | 9 | ( 83) | | 9 | ( 96) | | 9 | ( 99) |
| | 10 | ( 39) | | 10 | ( 86) | | 10 | (131) |
| | 11 | ( 61) | | 11 | (112) | | 11 | (105) |
| | 12 | (125) | | 12 | (119) | | 12 | (184) |
| 76 | 1 | ( 66) | 79 | 1 | ( 50) | 82 | 1 | ( 29) |
| | 2 | ( 31) | | 2 | ( 32) | | 2 | ( 21) |
| | 3 | ( 95) | | 3 | ( 20) | | 3 | ( 9) |
| | 4 | ( 68) | | 4 | (109) | | 4 | ( 29) |
| | 5 | ( 79) | | 5 | (100) | | 5 | ( 5) |
| | 6 | ( 97) | | 6 | (178) | | | |
| | 7 | (135) | | 7 | (134) | | | |
| | 8 | ( 70) | | 8 | (183) | | | |
| | 9 | ( 77) | | 9 | ( 80) | | | |
| | 10 | ( 45) | | 10 | ( 97) | | | |
| | 11 | ( 6) | | 11 | (117) | | | |
| | 12 | ( 24) | | 12 | ( 99) | | | |
| 77 | 3 | ( 26) | 80 | 1 | (131) | | | |
| | 4 | ( 51) | | 2 | ( 43) | | | |
| | 5 | ( 5) | | 3 | ( 29) | | | |
| | 6 | (123) | | 4 | ( 50) | | | |
| | 7 | ( 67) | | 5 | ( 75) | | | |
| | 8 | (222) | | 6 | (235) | | | |
| | 9 | ( 90) | | 7 | (142) | | | |
| | 10 | (136) | | 8 | (207) | | | |
| | 11 | ( 23) | | 9 | (102) | | | |
| | 12 | (150) | | 10 | (107) | | | |
| | | | | 11 | (108) | | | |
| | | | | 12 | (109) | | | |

The steps used to create the Entrained.Benthos data bank are shown in the following flow diagram:




Entrained.Benthos Data Bank

The original entrained benthos data are stored in an internal MIDAS file as ENTRAINMENT-REV. This file was first converted to an MTS line file and then changed to a flat file. The detailed procedures for creating a line file from an internal MIDAS file was discussed previously. The similar procedures for creating a line file for entrained benthos are shown below:

```
#RUN STAT:MIDAS
(reading all the variables)
?READ INTERNAL FILE=ENTRAINMENT-REV VARIABLE=ALL
(writing the required variables with the desired order)
?WRITE VARIABLE=1-7,9-16,18-22,24-25 FILE=EINVMIDAS CASES=ALL
```

```
FORMAT=(F4.0,1X,6(F3.0,1X),F6.2,1X,34(F10.5,1X))
?FINISH
```

The order of the parameters in the EINVMIDAS is as follows: YEAR, MONTH, WEEK, PERIOD, PUMP, REPLIC, CUBICM, and 34 parameters for species.

## Reformat Program

Each species in the entrained benthos file EINVMIDAS is treated as a parameter in the same way as that in the lake benthos file. The program BEARRAN was written for the same purposes as those for the lake benthos data, to reduce the space reserved for those species that show no counts and to rearrange the data by placing one species per line in a flat file. The content of this program is listed in Table 4.22. The result from program BEARRAN was saved in a file named BEREMIDAS.

## Taxir Create Program

A Taxir Create program, BETAXIRCR, was used to create the Entrained.Benthos data bank. This computer data bank is saved at position 6 of the COOK tape. The contents of BETAXIRCR are listed in Table 4.23.

TABLE 4.22. Program BEARRAN.

```
C PROGRAM BEARRAN IS USED TO ORGANIZE  ENTRAINMENT BENTHOS DATA SETS
C FOR THE PERIOD OF 1975 TO 1982.  THE OUTPUT FILE IS IN THE FORM OF
C FLAT FILE WHICH LATER CAN BE ADDED TO ENTRAINED.BENTHOS DATA BANK.
C UNIT 5 IS THE INPUT FILE.
C UNIT 6 IS THE OUTPUT FILE. (FLAT FILE).
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      REAL SAMP,YEAR,MONTH,WEEK,PERID,PUMP,REP,CUBICM,SPCONT(34)
      REAL*8 SPNAME(34)
      INTEGER ISAMP,IYEAR,IMONTH,IWEEK,IPERID,IPUMP,IREP
      INTEGER*2 GROUPN(34)
C
C*******
C*******        INITIALIZE DATA.
C*******
C
      DATA SPNAME/8H P.HOYI1,8H P.HOYI2,8H P.HOYI3,8H P.HOYI4,
     18HP.HOYI G,8HP.HOYI S,8HP.HOYI M,8H T MYSIS,8H  T GAMM,
     18H T HYALL,8HCRANGONX,8H T ASELL,8H  T CHIR,8H  T NAID,
     18H T TUBIF,8H T STYLO,8H T ENCHY,8H T HIRUD,8H TGASTRO,
     18HT SPHAER,8H T PISID,8HT HYDRAC,8H T HYDRA,8H  T TURB,
     18HCRAYFISH,8H  OTHERS,8H TRICHOP,8H   EPHEM,8H ODONATA,
     18H COLEOPT,8H CHAOBOR,8H CULICID,8HSIMULIIO,8HOINSECTA/
      DATA    GROUPN/7*2HAM,2HMY,3*2HAM,2HAS,2HCH,4*2HOL,2HHI,
     12HGA,2HSP,2HPI,3*2HOT,2HCR,2HOT,8*2HOI/
C
C*******
C*******        READ DATA LINES CONTAINING 34 SPECIES COUNTS.
C*******
C
  100 READ(5,10,END=99) SAMP,YEAR,MONTH,WEEK,PERIO,PUMP,REP,
     1CUBICM,(SPCONT(I),I=1,34)
  10  FORMAT(F4.0,1X,6(F3.0,1X),F6.2,1X,34(F10.5,1X))
      ISAMP=SAMP
      IYEAR=YEAR
      IMONTH=MONTH
      IWEEK=WEEK
      IPERIO=PERIO
      IPUMP=PUMP
      IREP=REP
C
C*******
C*******        CHANGE YEAR CODES. (EX: CHANGE 1 TO 75).
C*******
C
      IF(IYEAR.EQ.1) IYEAR=75
      IF(IYEAR.EQ.2) IYEAR=76
      IF(IYEAR.EQ.3) IYEAR=77
      IF(IYEAR.EQ.4) IYEAR=78
      IF(IYEAR.EQ.5) IYEAR=79
      IF(IYEAR.EQ.6) IYEAR=80
      IF(IYEAR.EQ.7) IYEAR=81
      IF(IYEAR.EQ.8) IYEAR=82
C
C*******
C*******        WRITE THE FLAT FILE WITH THE SPECIFICATIONS
C*******        OF ONE SPECIES AT EACH LINE.
C*******
C
      DO 20 I=1,34
      IF(SPCONT(I).LE.0.0) GO TO 20
      WRITE(6,30) ISAMP,IYEAR,IMONTH,IWEEK,IPERID,IPUMP,
     1IREP,CUBICM,GROUPN(I),SPNAME(I),SPCONT(I)
  30  FORMAT(I3,3X,6(I2,3X),F6.2,3X,A2,3X,A8,3X,F10.5)
  20  CONTINUE
      GO TO 100
C
  99  STOP
      END
```

TABLE 4.23. Program BETAXIRCR.

```
RUN *TAXIR
CREATE ENTRAINED.BENTHOS,
SAMPLE(FROM 100 TO 999),
YEAR(FROM 70 TO 85),
MONTH(FROM 1 TO 12),
WEEK(FROM 1 TO 5),
PERIOD(FROM 1 TO 9),
PUMP(FROM 1 TO 2),
REPLIC(FROM 1 TO 2),
CUBIC M(FROM 0.00 TO 999.99),
GROUP(ORDER,AM,AS,CH,CR,GA,HI,OI,OL,OT,MY,PI,SP),
CODE(NAME),
CELLS(FROM 0.00000 TO 9999.99999)*
ENTER DATA LOCATION=BEREMIDAS, FORMAT=FIXED,
SAMPLE<1-3>,
YEAR<7-8>,
MONTH<12-13>,
WEEK<17-18>,
PERIOD<22-23>,
PUMP<27-28>,
REPLIC<32-33>,
CUBIC M<37-42>,
GROUP<46-47>,
CODE<51-58>,
CELLS<62-71>*
SAVE
STOP
```

IMPINGED BENTHOS

The Impinged.Benthos data bank contains 4,103 items and 11 descriptors, which are listed below:

1--YEAR

2--MONTH

3--PERIOD

4--CASE

5--NAME

6--SEX

7--REPRODUCTION

8--SIZE

9--NUMBER

10--TOTAL NUMBER

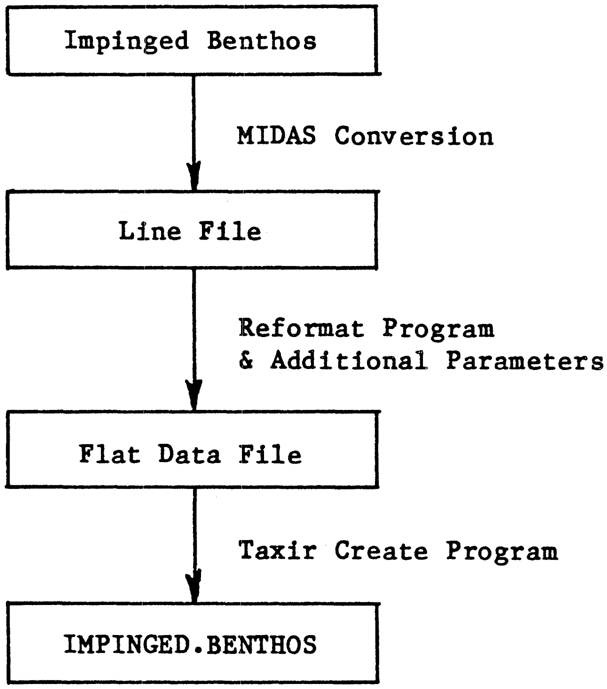
11--TOTAL WEIGHT

The monthly number of data items stored in the Impinged.Benthos data bank is shown in Table 4.24.

TABLE 4.24. The number of data items in Impinged.Benthos data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 75 | 1 | ( 13) | 78 | 1 | (27) | 81 | 1 | (34) |
| | 2 | ( 26) | | 2 | (19) | | 2 | (12) |
| | 3 | ( 51) | | 3 | (26) | | 3 | (40) |
| | 4 | (104) | | 4 | (53) | | 4 | (34) |
| | 5 | (101) | | 5 | (10) | | 6 | (10) |
| | 6 | ( 46) | | 6 | (17) | | 7 | (56) |
| | 7 | ( 18) | | 7 | (28) | | 8 | (49) |
| | 8 | (638) | | 8 | (47) | | 9 | (14) |
| | 9 | (525) | | 9 | (17) | | 10 | (11) |
| | 10 | (150) | | 10 | (19) | | 11 | (15) |
| | 11 | (120) | | 11 | (19) | | 12 | (14) |
| | 12 | ( 39) | | 12 | ( 4) | | | |
| 76 | 1 | ( 57) | 79 | 1 | ( 4) | | | |
| | 2 | ( 30) | | 2 | ( 4) | | | |
| | 3 | ( 36) | | 3 | (18) | | | |
| | 4 | ( 57) | | 4 | ( 9) | | | |
| | 5 | ( 22) | | 5 | ( 2) | | | |
| | 6 | ( 48) | | 7 | (41) | | | |
| | 7 | (102) | | 8 | (49) | | | |
| | 8 | (102) | | 9 | (32) | | | |
| | 9 | ( 81) | | 10 | (16) | | | |
| | 10 | ( 41) | | 11 | (16) | | | |
| | 11 | ( 38) | | 12 | ( 2) | | | |
| | 12 | ( 33) | 80 | 2 | (16) | | | |
| 77 | 1 | ( 11) | | 3 | (38) | | | |
| | 2 | ( 9) | | 4 | (59) | | | |
| | 3 | ( 39) | | 5 | (66) | | | |
| | 4 | ( 94) | | 6 | (12) | | | |
| | 5 | ( 56) | | 7 | (15) | | | |
| | 6 | ( 69) | | 8 | (63) | | | |
| | 7 | (110) | | 9 | (31) | | | |
| | 8 | ( 45) | | 10 | (21) | | | |
| | 9 | ( 29) | | 11 | ( 2) | | | |
| | 10 | ( 24) | | 12 | (13) | | | |
| | 11 | ( 18) | | | | | | |
| | 12 | ( 50) | | | | | | |

The steps for establishing this data bank are shown in the following flow diagram:




Impinged.Benthos Data Bank

The data for impinged benthos are also in an internal MIDAS file. The same MTS procedures used for both lake and entrained benthos are used here. The computer statements for the operations are listed below:

```
#RUN STAT:MIDAS
(read all the variables)
?READ INTERNAL FILE=IMPINGE VARIABLE=ALL
(write the required parameters)
?WRITE VARIABLE=1-44,46-49 FILE=IINVMIDAS CASES=ALL
FORMAT=(3(F3.0,1X),45(F7.1,1X))
?FINISH
```

The order of the original variables in IINVMIDAS is as follows: YEAR, MONTH, REPRODUCTION, SPECIES (the parameters of 43 species), TOTAL NUMBER, TOTAL WEIGHT.

## Reformat Program

The reformat program BIARRAN was used to reduce the space occupied by those impinged benthos species that show no counts and to rearrange the data so that each species is in the form of one species per line in a flat file. The contents of BIARRAN are listed in Table 4.25.

## Taxir Create Program

The Taxir Create program BITAXIRCR was used to create the Impinged.Benthos data bank. The complete Impinged.Benthos data bank is stored on tape COOK at position 7. The contents of the program are listed in Table 4.26.

TABLE 4.25. Program BIARRAN.

```
C PROGRAM BIARRAN IS USED TO ORGANIZE IMPINGEMENT BENTHOS DATA SETS
C FOR THE PERIOD OF 1975 TO 1981. THE OUTPUT FILE IS IN THE FORM OF
C FLAT FILE WHICH LATER CAN BE ADDED TO IMPINGED.BENTHOS DATA BANK.
C UNIT 5 IS ASSIGNED TO INPUT FILE.
C UNIT 6 IS THE OUTPUT FILE, (FLAT FILE).
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      REAL YEAR(815),MONTH(815),PERIOD(815),SPCONT(815,43),
     1TOTALN(815),TOTALW(815)
      REAL*8 NAME(43)
      INTEGER IYEAR(815),IMONTH(815),IPERID(815)
      INTEGER*2 SEX(43),REPROD(43),SIZE(43)
C
C*******
C*******        INITIALIZE DATA.
C*******
C
      DATA NAME/41*8HORC PROP,8HMUTILATE,8HOTHER SP/
      DATA  SEX/2*2H F,2HM1,2HM2,2*2H F,2HM1,2HM2,2*2H F,
     12HM1,2HM2,2*2H F,2HM1,2HM2,2*2H F,2HM1,2HM2,2*2H F,
     12HM1,2HM2,2*2H F,2HM1,2HM2,2*2H F,2HM1,2HM2,2*2H F,
     12HM1,2HM2,2*2H F,2HM1,2HM2,2H M,2*2H  /
C
      DATA REPROD/2H R,2HNR,2*2H  ,2H R,2HNR,2*2H  ,2H R,
     12HNR,2*2H  ,2H R,2HNR,2*2H  ,2H R,2HNR,2*2H  ,2H R,
     12HNR,2*2H  ,2H R,2HNR,2*2H  ,2H R,2HNR,2*2H  ,2H R,
     12HNR,2*2H  ,2H R,2HNR,5*2H  /
      DATA SIZE/4*2H 1,4*2H 2,4*2H 3,4*2H 4,4*2H 5,4*2H 6,
     14*2H 7,4*2H 8,4*2H 9,4*2H10,3*2H  /
C
C*******
C*******        READ DATA LINES CONTAINING 43 SPECIES COUNTS.
C*******
C
      DO 10 I=1,815
      READ(5,20) YEAR(I),MONTH(I),PERIOD(I),(SPCONT(I,J),J=1,43),
     1TOTALN(I),TOTALW(I)
 20   FORMAT(3(F3.0,1X),45(F7.1,1X))
      IYEAR(I)=YEAR(I)
      IMONTH(I)=MONTH(I)
      IPERID(I)=PERIOD(I)
C
C*******
C*******        CHANGE YEAR CODES, (EX: CHANGE 1 TO 75).
C*******
C
      IF(IYEAR(I).EQ.1) IYEAR(I)=75
      IF(IYEAR(I).EQ.2) IYEAR(I)=76
      IF(IYEAR(I).EQ.3) IYEAR(I)=77
      IF(IYEAR(I).EQ.4) IYEAR(I)=78
      IF(IYEAR(I).EQ.5) IYEAR(I)=79
      IF(IYEAR(I).EQ.6) IYEAR(I)=80
      IF(IYEAR(I).EQ.7) IYEAR(I)=81
C
C*******
C*******        WRITE THE FLAT FILE WITH THE SPECIFICATIONS
C*******        OF ONE SPECIES AT EACH LINE.
C*******
C
      DO 30 K=1,43
      IF(SPCONT(I,K).LE.0.) GO TO 30
      WRITE(6,40) IYEAR(I),IMONTH(I),IPERID(I),I,NAME(K),
     1SEX(K),REPROD(K),SIZE(K),SPCONT(I,K),TOTALN(I),TOTALW(I)
 40   FORMAT(3(I2,3X),I3,3X,A8,3(3X,A2),3(3X,F7.1))
 30   CONTINUE
 10   CONTINUE
C
      STOP
      END
```

TABLE 4.26. Program BITAXIRCR.

```
RUN *TAXIR
CREATE IMPINGED.BENTHOS,
YEAR(FROM 70 TO 85),
MONTH(FROM 1 TO 12),
PERIOD(FROM 1 TO 9),
CASE(FROM 1 TO 999),
NAME(NAME),
SEX(NAME),
REPROD(NAME),
SIZE(FROM 1 TO 10),
NUMBER(FROM 0.0 TO 99999.9),
TOTALN(FROM 0.0 TO 99999.9),
TOTALW(FROM 0.0 TO 99999.9)*
ENTER DATA LOCATION=BIREMIDAS, FORMAT=FIXED,
YEAR<1-2>,
MONTH<6-7>,
PERIOD<11-12>,
CASE<16-18>,
NAME<22-29>,
SEX<33-34>,
REPROD<38-39>,
SIZE<43-44>,
NUMBER<48-54>,
TOTALN<58-64>,
TOTALW<68-74>*
SAVE
STOP
```

## SUMMARY STATISTICS FOR ADULT LAKE AND IMPINGED FISH

This Adult.Fish.Summary.Statistics data bank is one of the largest in the Cook project. It consists of 102,238 items and 10 descriptors. The descriptors are listed below:

1-SPECIES

2-MONTH

3-YEAR

4-GEAR

5-STATION

6-SERIES

7-TEMPERATURE

8-INTERVAL

9-TOTAL NUMBER

10-TOTAL WEIGHT

The monthly number of data items stored in the Adult.Fish.Summary.Statistics data bank is listed in Table 4.27.

TABLE 4.27. The number of data items in the Adult.Fish.Summary.Statistics data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 73 | 1 | (25) | 76 | 1 | (992) | 79 | 1 | (261) |
| | 2 | (127) | | 2 | (728) | | 2 | (87) |
| | 3 | (282) | | 3 | (569) | | 3 | (267) |
| | 4 | (847) | | 4 | (1,125) | | 4 | (1,227) |
| | 5 | (922) | | 5 | (1,314) | | 5 | (788) |
| | 6 | (1,257) | | 6 | (997) | | 6 | (1,042) |
| | 7 | (891) | | 7 | (1,652) | | 7 | (1,131) |
| | 8 | (1,126) | | 8 | (1,006) | | 8 | (1,177) |
| | 9 | (792) | | 9 | (1,016) | | 9 | (1,445) |
| | 10 | (870) | | 10 | (761) | | 10 | (1,000) |
| | 11 | (145) | | 11 | (316) | | 11 | (752) |
| | 12 | (68) | | 12 | (233) | | 12 | (112) |
| 74 | 1 | (114) | 77 | 1 | (51) | 80 | 1 | (62) |
| | 2 | (21) | | 2 | (67) | | 2 | (149) |
| | 3 | (445) | | 3 | (257) | | 3 | (185) |
| | 4 | (805) | | 4 | (507) | | 4 | (669) |
| | 5 | (1,160) | | 5 | (936) | | 5 | (1,562) |
| | 6 | (867) | | 6 | (951) | | 6 | (1,681) |
| | 7 | (1,564) | | 7 | (1,115) | | 7 | (1,006) |
| | 8 | (1,097) | | 8 | (891) | | 8 | (1,141) |
| | 9 | (725) | | 9 | (1,088) | | 9 | (1,496) |
| | 10 | (584) | | 10 | (936) | | 10 | (1,288) |
| | 11 | (326) | | 11 | (748) | | 11 | (730) |
| | 12 | (165) | | 12 | (179) | | 12 | (491) |
| 75 | 1 | (285) | 78 | 1 | (150) | 81 | 1 | (307) |
| | 2 | (215) | | 2 | (56) | | 2 | (143) |
| | 3 | (791) | | 3 | (141) | | 3 | (123) |
| | 4 | (2,843) | | 4 | (412) | | 4 | (959) |
| | 5 | (2,698) | | 5 | (718) | | 5 | (1,562) |
| | 6 | (3,286) | | 6 | (1,128) | | 6 | (1,722) |
| | 7 | (1,872) | | 7 | (1,359) | | 7 | (1,617) |
| | 8 | (1,875) | | 8 | (1,368) | | 8 | (840) |
| | 9 | (1,580) | | 9 | (1,140) | | 9 | (966) |
| | 10 | (1,682) | | 10 | (1,373) | | 10 | (1,176) |
| | 11 | (2,279) | | 11 | (724) | | 11 | (1,233) |
| | 12 | (2,102) | | 12 | (222) | | 12 | (550) |
| | | | | | | 82 | 1 | (250) |
| | | | | | | | 2 | (90) |
| | | | | | | | 3 | (128) |
| | | | | | | | 4 | (1,147) |
| | | | | | | | 5 | (908) |
| | | | | | | | 6 | (1,131) |
| | | | | | | | 7 | (875) |
| | | | | | | | 8 | (709) |
| | | | | | | | 9 | (817) |
| | | | | | | | 10 | (584) |
| | | | | | | | 11 | (602) |
| | | | | | | | 12 | (91) |

The procedures for establishing this data bank are shown as follows:

Adult Fish Summary
Statistics Files

↓ Taxir Create Program

ADULT.FISH.SUMMARY.
STATISTICS Data Bank

Adult.Fish.Summary.Statistics Data Bank

The original data for adult lake and impinged fish summary statistics are stored in flat-file form; therefore, no procedure is necessary to rearrange this data format. A Taxir Create program was used to create the Adult.Fish. Summary.Statistics data bank directly from the original data files. These computer data are stored on tape COOK at position 8. The Taxir Create program AFSSTAXIRCR is shown in Table 4.28.

TABLE 4.28. Program AFSSTAXIRCR.

```
RUN *TAXIR
CREATE ADULT.FISH.SUMMARY.STATISTICS,
SPECIES(FROM 0 TO 99),
MONTH(FROM 1 TO 12),
YEAR(FROM 70 TO 85),
GEAR(FROM 0 TO 99),
STATION(FROM 0 TO 99),
SERIES(FROM 0 TO 99),
TEMPERATURE(FROM 0.0 TO 99.9),
INTERVAL(FROM 0 TO 999),
TOTALNUMBER(FROM 0 TO 999999),
TOTALWEIGHT(FROM 0.0 TO 99999999.9)*
ENTER DATA LOCATION=MASTERFILE, FORMAT=FIXED,
SPECIES<1-2>,
MONTH<3-4>,
YEAR<5-6>,
GEAR<7-8>,
STATION<9-10>,
SERIES<11-12>,
TEMPERATURE<13-16>,
INTERVAL<17-19>,
TOTALNUMBER<20-25>,
TOTALWEIGHT<26-35>*
SAVE
STOP
```

## FIELD-CAUGHT AND IMPINGED FISH

The data for field-caught and impinged fish constitute the largest data set in the Cook project. Because the data are too extensive to be held in a single file, the data were divided into two files. The first one contains the raw data on adult lake fish stored in computer data base Lake.Adult.Fish; the second one includes the raw data on impinged adult fish stored in the computer data base Impinged.Adult.Fish. Both data banks have 22 descriptors:

| | |
|---|---|
| 1-MONTH | 12-SPECIES |
| 2-DAY | 13-FISH NO. |
| 3-YEAR | 14-LENGTH |
| 4-TIME | 15-WEIGHT |
| 5-GEAR | 16-SEX |
| 6-SERIES | 17-GONAD COND. |
| 7-STATIC | 18-GILLNET HR. |
| 8-WATER TEMP. | 19-GILLNET MIN. |
| 9-FISHUSE | 20-FOOD PRESENT |
| 10-BIOL.COND. | 21-SUBSAMPLE |
| 11-PHY.COND. | 22-TWNONSUB |

The Lake.Adult.Fish data bank has 181,156 items, and the Impinged.Adult. Fish data bank includes 110,843 items. The numbers of data items stored in the data banks are shown in Tables 4.29 and 4.30.

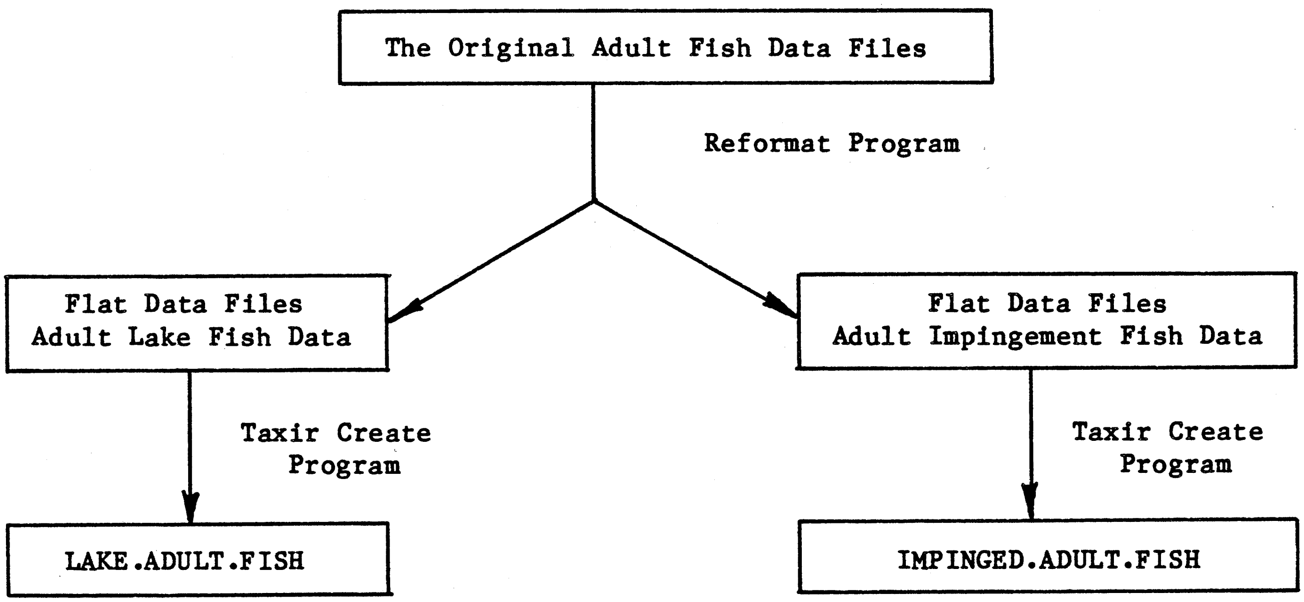
TABLE 4.29. The number of data items in Lake.Adult.Fish data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 73 | 2 | ( 80) | 76 | 2 | ( 143) | 80 | 4 | ( 953) |
| | 3 | ( 830) | | 3 | ( 205) | | 5 | (4,508) |
| | 4 | (2,611) | | 4 | (2,146) | | 6 | (3,819) |
| | 5 | (2,445) | | 5 | (2,700) | | 7 | (2,957) |
| | 6 | (3,746) | | 6 | (2,008) | | 8 | (2,411) |
| | 7 | (2,219) | | 7 | (3,302) | | 9 | (3,738) |
| | 8 | (3,130) | | 8 | (2,117) | | 10 | (2,887) |
| | 9 | (1,568) | | 9 | (2,153) | | 11 | (1,725) |
| | 10 | (2,263) | | 10 | (1,448) | 81 | 4 | (2,324) |
| | 11 | ( 211) | | 11 | ( 326) | | 5 | (2,973) |
| | 12 | ( 7) | 77 | 3 | ( 206) | | 6 | (4,264) |
| 74 | 1 | ( 82) | | 4 | ( 413) | | 7 | (4,443) |
| | 3 | ( 419) | | 5 | (2,057) | | 8 | (1,275) |
| | 4 | (1,866) | | 6 | (1,947) | | 9 | (1,769) |
| | 5 | (3,516) | | 7 | (2,554) | | 10 | (2,418) |
| | 6 | (2,448) | | 8 | (2,205) | | 11 | (2,099) |
| | 7 | (3,177) | | 9 | (2,646) | 82 | 4 | (1,663) |
| | 8 | (2,256) | | 10 | (2,063) | | 5 | (1,711) |
| | 9 | (1,526) | | 11 | (1,494) | | 6 | (2,128) |
| | 10 | (1,284) | | 12 | ( 40) | | 7 | (1,433) |
| | 11 | ( 567) | 78 | 4 | ( 434) | | 8 | (1,621) |
| | 12 | ( 91) | | 5 | (1,313) | | 9 | (2,258) |
| 75 | 1 | ( 94) | | 6 | (2,854) | | 10 | ( 896) |
| | 3 | ( 376) | | 7 | (3,417) | | 11 | (1,186) |
| | 4 | ( 703) | | 8 | (2,976) | | | |
| | 5 | (2,319) | | 9 | (2,175) | | | |
| | 6 | (3,156) | | 10 | (3,227) | | | |
| | 7 | (2,055) | | 11 | (1,313) | | | |
| | 8 | (1,753) | 79 | 4 | (2,399) | | | |
| | 9 | (2,495) | | 5 | (1,934) | | | |
| | 10 | (1,541) | | 6 | (3,004) | | | |
| | 11 | (1,208) | | 7 | (2,967) | | | |
| | 12 | ( 917) | | 8 | (2,647) | | | |
| | | | | 9 | (3,509) | | | |
| | | | | 10 | (2,677) | | | |
| | | | | 11 | (1,719) | | | |

TABLE 4.30. The number of data items in the Impinged.Adult.Fish data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 73 | 1 | (33) | 77 | 1 | (68) | 80 | 1 | (132) |
| | 2 | (157) | | 2 | (118) | | 2 | (383) |
| | 3 | (11) | | 3 | (456) | | 3 | (572) |
| | 4 | (99) | | 4 | (588) | | 4 | (2,667) |
| | 10 | (2) | | 5 | (337) | | 5 | (1,667) |
| | 11 | (6) | | 6 | (325) | | 6 | (2,159) |
| | 12 | (140) | | 7 | (521) | | 7 | (488) |
| 74 | 1 | (98) | | 8 | (80) | | 8 | (1,384) |
| | 2 | (30) | | 9 | (148) | | 9 | (2,260) |
| | 3 | (745) | | 10 | (622) | | 10 | (1,530) |
| | 4 | (255) | | 11 | (326) | | 11 | (51) |
| | 5 | (173) | | 12 | (316) | | 12 | (1,294) |
| | 6 | (38) | 78 | 1 | (305) | 81 | 1 | (1,247) |
| | 7 | (1,193) | | 2 | (73) | | 2 | (284) |
| | 8 | (767) | | 3 | (376) | | 3 | (256) |
| | 9 | (276) | | 4 | (333) | | 4 | (708) |
| | 10 | (52) | | 5 | (470) | | 5 | (3,466) |
| | 11 | (122) | | 6 | (408) | | 6 | (2,231) |
| | 12 | (205) | | 7 | (1,235) | | 7 | (1,326) |
| 75 | 1 | (467) | | 8 | (1,286) | | 8 | (983) |
| | 2 | (424) | | 9 | (468) | | 9 | (519) |
| | 3 | (1,063) | | 10 | (628) | | 10 | (997) |
| | 4 | (8,084) | | 11 | (388) | | 11 | (1,754) |
| | 5 | (4,853) | | 12 | (484) | | 12 | (2,606) |
| | 6 | (6,153) | 79 | 1 | (719) | 82 | 1 | (565) |
| | 7 | (2,097) | | 2 | (130) | | 2 | (206) |
| | 8 | (1,735) | | 3 | (514) | | 3 | (262) |
| | 9 | (1,344) | | 4 | (661) | | 4 | (2,140) |
| | 10 | (3,879) | | 5 | (45) | | 5 | (1,575) |
| | 11 | (4,189) | | 6 | (5) | | 6 | (1,872) |
| | 12 | (3,697) | | 7 | (1,053) | | 7 | (1,032) |
| 76 | 1 | (2,626) | | 8 | (1,027) | | 8 | (296) |
| | 2 | (1,249) | | 9 | (1,359) | | 9 | (134) |
| | 3 | (1,611) | | 10 | (814) | | 10 | (384) |
| | 4 | (574) | | 11 | (124) | | 11 | (562) |
| | 5 | (759) | | 12 | (276) | | 12 | (228) |
| | 6 | (616) | | | | | | |
| | 7 | (1,099) | | | | | | |
| | 8 | (548) | | | | | | |
| | 9 | (575) | | | | | | |
| | 10 | (593) | | | | | | |
| | 11 | (378) | | | | | | |
| | 12 | (522) | | | | | | |

The procedures for creating these two data banks are shown in the flow diagram below:




Reformat Program

The program CHNGFRMT was used to reorganize the data for the adult lake and impinged fish which are stored in a file named TRANSREC. It separates lake fish data from the impingement data and enters the results of this program in flat-file forms stored in REFTRANFIE and REFTRANIMP, where the former contains the lake data and the latter houses the impingement data. These procedures are shown below:

```
#RUN *FTN SCARDS=CHNGFRMT SPUNCH=C.OBJ
#RUN C.OBJ 5=TRANSREC 6=REFTRANFIE 7=REFTRANIMP
```

(Table 4.31 is a copy of program CHNGFRMT)

TABLE 4.31. Program CHNGFRMT.

```
C PROGRAM CHNGFRMT  IS USED TO SEPARATE FIELD RECORDS FROM THE IMPINGEMENT
C RECORDS IN THE TRANSREC FILES. THE OUTPUT FILES PROVIDED BY THIS PROGRAM
C ARE IN THE FORM OF FLAT FILES.
C UNIT 5 IS THE INPUT FILE, (TRANSREC FILES).
C UNIT 6 IS THE OUTPUT FILE FOR IMPINGEMENT RECORDS.
C UNIT 7 IS THE OUTPUT FILE FOR FIELD RECORDS.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      INTEGER AB
      REAL D,E,H
      LOGICAL*1 A(13),AC(7),B(20),C(18)
C
C*******
C*******         READ THE DATA LINES FROM TRANSREC FILES.
C*******
C
 100  READ(5,1,END=99) A,AB,AC,D,B,E,C,H
 1    FORMAT(13A1,I1,7A1,F4.1,20A1,F5.0,18A1,F7.0)
C
C*******
C*******         CHECK THE TYPE OF THE RECORD, (IMPINGEMENT OR FIELD).
C*******
C
      IF(H.EQ.0.0) GO TO 10
      IF(AB.EQ.1) GO TO 20
C
C*******
C*******         WRITE THE IMPINGEMENT AND FIELD RECORDS.
C*******
C
      WRITE(6,2) A,AB,AC,D,B,E,C,H
 2    FORMAT(13A1,I1,7A1,F4.1,20A1,F8.2,18A1,F9.1)
      GO TO 100
C
 20   WRITE(7,2) A,AB,AC,D,B,E,C,H
      GO TO 100
C
 10   IF(AB.EQ.1) GO TO 30
      WRITE(6,3) A,AB,AC,D,B,E,C
 3    FORMAT(13A1,I1,7A1,F4.1,20A1,F8.2,18A1)
      GO TO 100
C
 30   WRITE(7,3) A,AB,AC,D,B,E,C
      GO TO 100
C
 99   STOP
      END
```

## Taxir Create Program

The Taxir Create program AFTAXIRCR (Table 4.32) was used to create adult lake and impinged fish data banks from the flat files REFTRANFIE and REFTRANIMP. The results of the Taxir Create program are saved in the Lake.Adult.Fish and Impinged.Adult.Fish data banks. The final version of Lake.Adult.Fish is saved on tape COOK at position 9, and that for Impinged.Adult.Fish is stored on the same tape at position 10.

TABLE 4.32. Program AFTAXIRCR.

```
RUN *TAXIR
CREATE IMPINGED.ADULT.FISH,
MONTH(FROM 1 TO 12),
DAY(FROM 1 TO 31),
YEAR(FROM 70 TO 85),
TIME(FROM 0 TO 2400),
GEAR(FROM 0 TO 9),
SERIES(FROM 0 TO 99),
STATION(FROM 0 TO 9),
WATERTEMP.(FROM 0.0 TO 99.9),
FISHUSE(FROM 0 TO 9),
BIOL.COND.(FROM 0 TO 9),
PHY.COND.(FROM 0 TO 9),
SPECIES(FROM 0 TO 99),
FISHNO.(FROM 0 TO 999999),
LENGTH(FROM 0 TO 9999),
WEIGHT(FROM 0.00 TO 99999.99),
SEX(FROM 0 TO 9),
GONADCOND.(FROM 0 TO 9),
GILLNETHR.(FROM 0 TO 99),
GILLNETMIN.(FROM 0 TO 99),
FOODPRESENT(FROM 0 TO 9),
SUBSAMPLE(FROM 0 TO 99),
TWNONSUB(FROM 0.0 TO 9999999.9)*
ENTER DATA LOCATION=REFTRANIMP, FORMAT=FIXED,
MONTH<1-2>,
DAY<3-4>,
YEAR<5-6>,
TIME<9-12>,
GEAR<14>,
SERIES<15-16>,
STATION<19>,
WATERTEMP.<22-25>,
FISHUSE<29>,
BIOL.COND.<30>,
PHY.COND.<31>,
SPECIES<32-33>,
FISHNO.<34-39>,
LENGTH<41-44>,
WEIGHT<46-53>,
SEX<56>,
GONADCOND.<59>,
GILLNETHR.<60-61>,
GILLNETMIN.<62-63>,
FOODPRESENT<64>,
SUBSAMPLE<68-69>,
TWNONSUB<72-80>*
SAVE
STOP
```

FIELD: LARVAL FISH

The Lake.Larvae data bank contains 20 descriptors and 14,110 items. The descriptors are as follows:

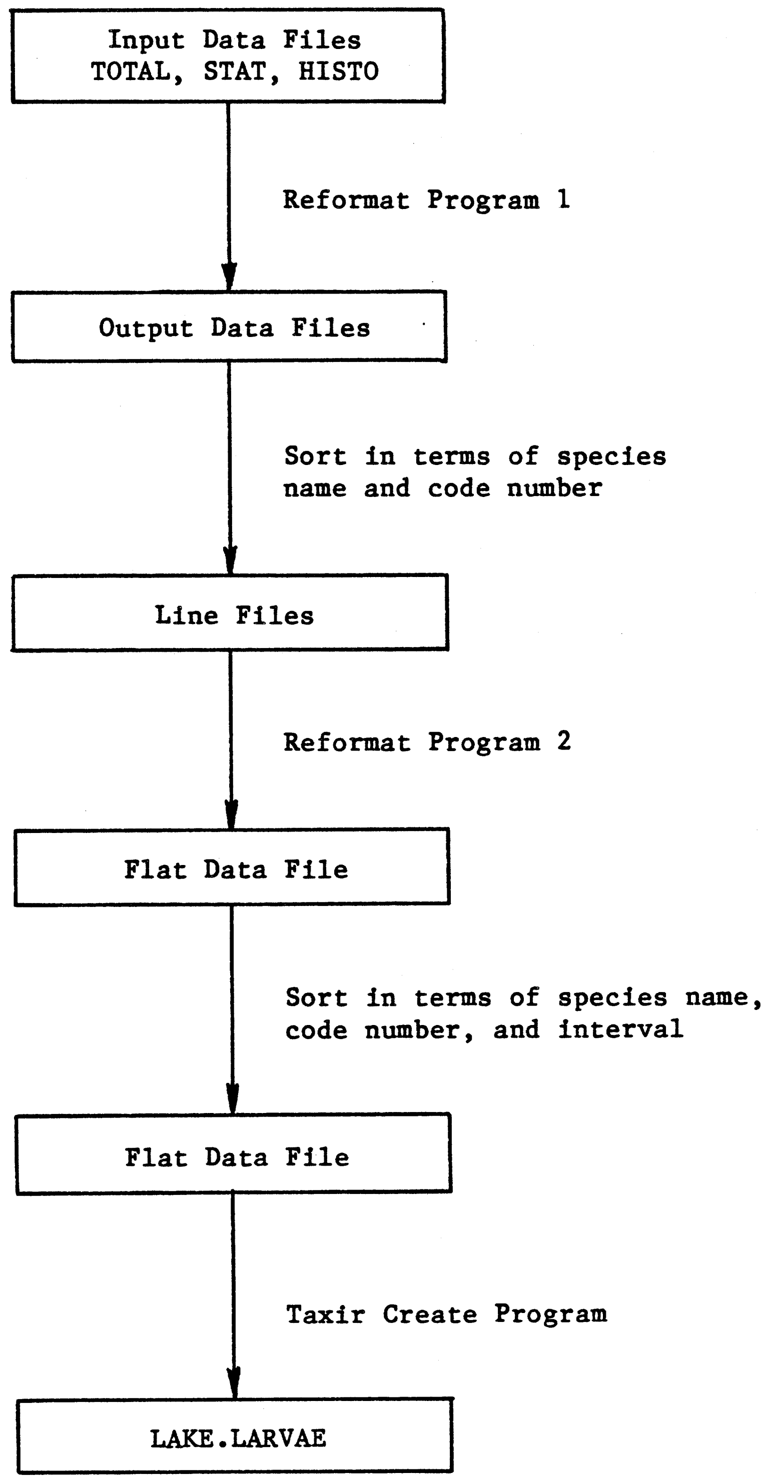
| | |
|---|---|
| 1-CODE NO. | 11-GRATE |
| 2-SAMPLE NO. | 12-STATION |
| 3-PERIOD | 13-TEMPERATURE |
| 4-MONTH | 14-REVOLATION |
| 5-DAY | 15-SPECIES NAME |
| 6-YEAR | 16-SPECIES DENSITY |
| 7-DIAL | 17-INTERVAL |
| 8-TIME | 18-NUMBER |
| 9-GEAR | 19-SUM LENGTH |
| 10-TOW DEPTH | 20-SUM SQUARE LENGTH |

The monthly number of data items stored in the Lake.Larvae data bank is listed in Table 4.33.

TABLE 4.33. The number of data items in Lake.Larvae data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 73 | 3 | ( 8) | 77 | 4 | ( 56) | 81 | 4 | ( 77) |
| | 4 | ( 80) | | 5 | ( 56) | | 5 | (139) |
| | 5 | ( 55) | | 6 | (213) | | 6 | (239) |
| | 6 | (404) | | 7 | (398) | | 7 | (184) |
| | 7 | (372) | | 8 | (311) | | 8 | (480) |
| | 8 | (151) | | 9 | ( 91) | | 9 | (111) |
| | 9 | ( 45) | | 10 | ( 17) | | 10 | ( 12) |
| | 10 | ( 52) | | 11 | ( 17) | | 11 | ( 12) |
| | 11 | ( 10) | 78 | 4 | ( 78) | 82 | 4 | ( 76) |
| 74 | 1 | ( 4) | | 5 | ( 86) | | 5 | (106) |
| | 3 | ( 12) | | 6 | (150) | | 6 | (483) |
| | 4 | ( 58) | | 7 | (121) | | 7 | (492) |
| | 5 | (171) | | 8 | (237) | | 8 | (466) |
| | 6 | (395) | | 9 | ( 74) | | 9 | ( 12) |
| | 7 | (463) | | 10 | ( 19) | | 10 | ( 12) |
| | 8 | (336) | | 11 | ( 13) | | 11 | ( 12) |
| | 9 | (155) | 79 | 4 | ( 72) | | | |
| | 10 | ( 58) | | 5 | ( 88) | | | |
| | 11 | ( 30) | | 6 | (125) | | | |
| 75 | 4 | ( 74) | | 7 | (507) | | | |
| | 5 | (120) | | 8 | (358) | | | |
| | 6 | (300) | | 9 | (122) | | | |
| | 7 | (521) | | 10 | ( 34) | | | |
| | 8 | (419) | | 11 | ( 12) | | | |
| | 9 | (102) | 80 | 4 | ( 73) | | | |
| | 10 | ( 80) | | 5 | (169) | | | |
| | 11 | ( 68) | | 6 | (124) | | | |
| 76 | 2 | ( 8) | | 7 | (743) | | | |
| | 4 | ( 73) | | 8 | (207) | | | |
| | 5 | ( 79) | | 9 | ( 96) | | | |
| | 6 | (299) | | 10 | ( 14) | | | |
| | 7 | (930) | | 11 | ( 13) | | | |
| | 8 | (439) | | | | | | |
| | 9 | ( 73) | | | | | | |
| | 10 | ( 47) | | | | | | |
| | 11 | ( 12) | | | | | | |

The flow chart below shows the different stages involved in the creation of the Lake.Larvae data bank.



The purpose of this operation is to combine the data in various formats in three files into one single flat file, which is then entered into the Taxir data base management program to create a data base management file. The original data are contained in three files called HISTOYR, STATYR, and TOTALYR, where "YR" indicates the year when the data were collected. Both the open lake and the nearshore beach data are included in these files. The HISTOYR file contains the following parameters: CODE NO., SAMPLE NO., PERIOD, DATE, DIAL, TIME, GEAR, TOW DEPTH, GRATE, STATION, TEMPERATURE, REVOLATION, SPECIES NAME, and 51 different density intervals. The first 12 parameters of the STATYR file are the same as those in HISTOYR file but the parameters which follow are SPECIES CODES and statistical values of TOTAL NUMBER, SUM, and SUM OF SQUARE for 20 species. The first 12 parameters of the TOTALYR file are also the same as those in the HISTOYR file, but are followed by the single parameter, total egg counts.

Reformat Programs

All three files are identical in the first 12 parameters but are different in the parameters which follow. Reformat programs are needed to rearrange these files, so that all parameters can be merged into one single flat file. Two reformat programs were used. The first reformat program is called LLARVARR, as shown in Table 4.34. The major functions of this program are 1) to change STATYR file from 20 species per line to one species per line, 2) to delete those lines with no density values in the HISTOYR file, and 3) to add the character "EG" to each line in TOTALYR file to indicate that these data are for fish eggs.

TABLE 4.34. Program LLARVARR.

```
C PROGRAM LLARVARR HAS THREE FUNCTIONS. IT CHANGES STAT FILE INTO A FLAT
C FILE, DELETES THE DATA LINES WITH NO HISTOGRAM VALUES FROM HISTO FILE,
C AND FINALLY IT ADDS CHARACTER "EG" TO EACH LINE OF TOTAL FILE.
C UNITS 2, 4, AND 6 ARE THE INPUT FILES.
C UNITS 3, 5, AND 7 ARE THE OUTPUT FILES.
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      INTEGER SPN(20),NUMI(20),SAMPNO,PRD,DATE,DIEL,TIME,GEAR,
     1TOWDEP,GRATE,STATN,SPECN,IREV,EGGCON
      REAL SUMI(20),SUMSI(20),TEMP,REV,ROUNDC(51),SPCON(20)
      REAL*8 CODENO
C
C*******
C*******        DELETE THE DATA LINES FROM STAT FILE WITH NO STATISTICS.
C*******        MAKE STAT FILE FLAT.
C*******
C
 100  READ(2,10,END=99) CODENO,(SPN(I),NUMI(I),SUMI(I),
     1SUMSI(I),I=1,20)
 10   FORMAT(A5,42X,20(1X,A2,1X,I4,1X,F9.1,1X,F11.1))
C
      IF(NUMI(1).EQ.0) GO TO 100
C
      DO 20 I=1,20
      IF(NUMI(I).EQ.0.) GO TO 20
      WRITE(3,30) CODENO,SPN(I),NUMI(I),SUMI(I),SUMSI(I)
 30   FORMAT(A5,1X,A2,1X,I4,1X,F9.1,1X,F11.2)
 20   CONTINUE
C
      GO TO 100
C
C*******
C*******        DELETE DATA LINES FROM HISTO FILE WITH NO HISTOGRAM VALUES.
C*******
C
 99   READ(4,11,END=999) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,
     1GEAR,TOWDEP,GRATE,STATN,TEMP,REV,SPECN,(ROUNDC(I),I=1,51)
 11   FORMAT(A5,1X,A4,1X,I2,1X,I6,1X,I1,1X,I4,1X,A1,1X,
     1A2,A1,A2,1X,F4.1,1X,F6.0,1X,A2,51F6.0)
C
      DO 21 I=1,51
    . IF(ROUNDC(I).EQ.0.) GO TO 21
      GO TO 101
 21   CONTINUE
C
      GO TO 99
C
 101  WRITE(5,11) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,GEAR,
     1TOWDEP,GRATE,STATN,TEMP,REV,SPECN,(ROUNDC(I),I=1,51)
C
      GO TO 99
C
C*******
C*******        ADD CHARACTER "EG" TO EACH LINE OF TOTAL FILE.
C*******
C
 999  READ(6,12,END=9999) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,
     1GEAR,TOWDEP,GRATE,STATN,TEMP,REV,EGGCON
 12   FORMAT(A5,1X,A4,1X,I2,1X,I6,1X,I1,1X,I4,1X,A1,1X,A2,A1,
     1A2,1X,F4.1,1X,F6.0,6X,I9)
      IREV=REV
      WRITE(7,31) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,GEAR,
     1TOWDEP,GRATE,STATN,TEMP,IREV,EGGCON
 31   FORMAT(A5,1X,A4,1X,I2,1X,I6,1X,I1,1X,I4,1X,A1,1X,A2,
     1A1,A2,1X,F4.1,1X,I6,1X,'EG',1X,I9)
      GO TO 999
C
 9999 STOP
      END
```

The second reformat program is called MERGE, as shown in Table 4.35. This program merges the three output files provided by the LLARVARR program. In order to use this program, the output files were first sorted by species name and code number. The sorted data were stored in files SRSTATYR, SRHISTOYR, and SRTOTALYR, respectively. The MERGE program was then used, first to copy the SRTOTALYR at the beginning of a new file, and then to combine every pair of lines (one each from SRHISTOYR and SRSTATYR) into a single line of parameters in the new file. The 51 density intervals, however, were changed to a single density interval per line. Thus, the new file is flat, containing all information for fish species with their statistical values and one density interval per line. The new file, called MERGEYR, was sorted again at the end of the operation.

This final sorted flat file named SMERGEYR was set as input to a Taxir Create program. The computer commands and statements used in this process for the 1981 field larval fish data are shown below:

```
#RUN *FIN SCARDS=LLARVARR SPUNCH=LLARV.OBJ
#RUN LLARV.OBJ 2=STAT81 4=HISTO81 6=TOTAL81
               3=RSTAT81 5=RHISTO81 7=RTOTAL81
(to sort the output files in terms of species name and code number)
#RUN *SORT PAR=SORT=CH,A,1,5,CH,A,7,2
INPUT=RSTAT81,U,35,35 OUTPUT=SRSTAT81,U,35,35 END
#RUN *SORT PAR=SORT=CH,A,1,5,CH,A,49,2
INPUT=RHISTO81,U,356,356 OUTPUT=SRHISTO81,U,356,356 END
#RUN *SORT PAR=SORT=CH,A,1,5
INPUT=RTOTAL81,U,64,64 OUTPUT=SRTOTAL81,U,64,64 END
```

TABLE 4.35. Program MERGE.

```
C PROGRAM MERGE IS USED TO MERGE SRSTAT AND SRHISTO FILE. THE OUTPUT
C OF THIS PROGRAM IS IN THE FORM OF FLAT FILE.
C UNITS 3, 5, AND 7 ARE THE INPUT FILES.
C UNIT 9 IS THE OUTPUT FILE, (FLAT FILE).
C
C
C
C
C*******
C*******        INITIALIZE VARIABLES.
C*******
C
      INTEGER NUMI,SAMPNO,PRD,DATE,DIEL,TIME,GEAR,TOWDEP,
     1GRATE,STATN,IREV,SPN,IROUND(51)
      REAL SUMI,SUMSI,TEMP,REV,ROUNDC(51)
      REAL*8 CODENO
C
C*******
C*******        WRITE THE SRTOTAL FILE IN THE BEGINNING OF THE
C*******        OUTPUT FILE.
C*******
C
      READ(3,10,END=99) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,GEAR,
     1TOWDEP,GRATE,STATN,TEMP,IREV,SPN,EGGCON
 10   FORMAT(A5,1X,A4,1X,I2,1X,I6,1X,I1,1X,I4,1X,A1,1X,A2,A1,
     1A2,1X,F4.1,1X,I6,1X,A2,1X,I9)
      WRITE(9,10) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,GEAR,TOWDEP,
     1GRATE,STATN,TEMP,IREV,SPN,EGGCON
C
C*******
C*******        READ ONE LINE FROM SRSTAT FILE AND ONE LINE
C*******        FROM SRHISTO FILE AT A TIME.
C*******
C
 99   READ(5,20,END=999) NUMI,SUMI,SUMSI
 20   FORMAT(9X,I4,1X,F9.1,1X,F11.2)
      READ(7,30,END=999) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,
     1GEAR,TOWDEP,GRATE,STATN,TEMP,REV,SPN,(ROUNDC(I),I=1,51)
 30   FORMAT(A5,1X,A4,1X,I2,1X,I6,1X,I1,1X,I4,1X,A1,1X,A2,A1,
     1A2,1X,F4.1,1X,F6.0,1X,A2,51F6.0)
      IREV=REV
C
C*******
C*******        WRITE THE MERGED LINES.
C*******
C
      DO 40 I=1,51
      IF(ROUNDC(I).EQ.0.) GO TO 40
      IROUND(I)=ROUNDC(I)
      WRITE(9,50) CODENO,SAMPNO,PRD,DATE,DIEL,TIME,GEAR,
     1TOWDEP,GRATE,STATN,TEMP,IREV,SPN,IROUND(I),I,
     1NUMI,SUMI,SUMSI
 50   FORMAT(A5,1X,A4,1X,I2,1X,I6,1X,I1,1X,I4,1X,A1,1X,A2,A1,A2,
     11X,F4.1,1X,I6,1X,A2,4X,I6,1X,I2,1X,I4,1X,F9.1,1X,F11.2)
 40   CONTINUE
C
      GO TO 99
C
 999  STOP
      END
```

(to produce MERGE81)

```
#RUN *FTN SCARDS=MERGE SPUNCH=MER.OBJ

#RUN MER.OBJ 3=SRSTAT81 5=SRHISTO81 7=SRTOTAL81 9=MERGE81
```

(to sort MERGE81 in terms of species name, code number, and Intervals)

```
#RUN *SORT PAR=SORT=CH,A,1,5,CH,A,49,2,CH,A,62,2 INPUT=MERGE81,U,90,90 OUTPUT=
SMERGE81,U,90,90 END
```

## Taxir Create Program

A Taxir Create program FLTAXIRCR is used to create the Lake.Larvae data bank from SMERGEYR files. This data bank is stored on the tape COOK at position 11. The contents of FLTAXIRCR are presented in Table 4.36.

TABLE 4.36. Program FLTAXIRCR.

```
R *TAXIR
CREATE  LAKE.LARVAE,
CODENO(NAME),
SAMPNO(NAME),
PERIOD(FROM 0 TO 99),
MONTH(FROM 1 TO 12),
DAY(FROM 1 TO 31),
YEAR(FROM 70 TO 85),
DIEL(FROM 0 TO 9),
TIME(FROM 0 TO 9999),
GEAR(NAME),
TOWDEP(NAME),
GRATE(NAME),
STATN(NAME),
TEMP(FROM 0.0 TO 99.9),
REV(FROM 0 TO 999999),
SPECN(NAME),
SPECDEN(FROM 0 TO 999999),
INTERVAL(FROM 0 TO 99),
NUM(FROM 0 TO 9999),
SUMLEN(FROM 0.0 TO 9999999.9),
SUMSLEN(FROM 0.00 TO 9999999.99)*
ENTER DATA LOCATION=SMERGE, FORMAT=FIXED,
CODENO<1-5>,
SAMPNO<7-10>,
PERIOD<12-13>,
MONTH<15-16>,
DAY<17-18>,
YEAR<19-20>,
DIEL<22>,
TIME<24-27>,
GEAR<29>,
TOWDEP<31-32>,
GRATE<33>,
STATN<34-35>,
TEMP<37-40>,
REV<42-47>,
SPECN<49-50>,
SPECDEN<55-60>,
INTERVAL<62-63>,
NUM<65-68>,
SUMLEN<70-78>,
SUMSLEN<80-90>*
SAVE
STOP
```

ENTRAINMENT: LARVAL FISH

The Entrained.Larvae data bank contains 12,111 items and 25 descriptors. The descriptors in this data bank are shown below:

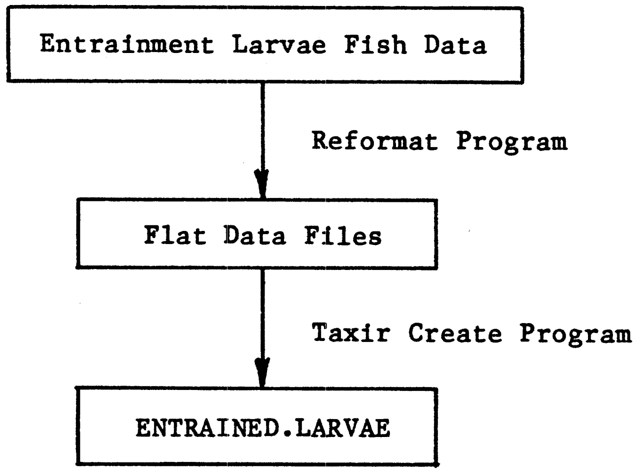
| | |
|---|---|
| 1-CASE | 14-DEPTH |
| 2-CODE | 15-STARTH |
| 3-SAMPLE NUMBER | 16-STARTM |
| 4-MONTH | 17-STOPH |
| 5-DAY | 18-STOPM |
| 6-YEAR | 19-TTIME |
| 7-J DAY | 20-DIEL |
| 8-MONTHPD | 21-TEMP |
| 9-GEAR | 22-REVS |
| 10-SERIES | 23-SPEC |
| 11-GRATE | 24-SPDEN |
| 12-NORTH/SOUTH | 25-INTERVAL |
| 13-INTAKE/DISCHARGE | |

The monthly number of data items stored in the Entrained.Larvae data bank is listed in Table 4.37.

TABLE 4.37. The number of data items stored in the Entrained.Larvae data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 75 | 1 | ( 8) | 78 | 1 | ( 27) | 81 | 1 | ( 32) |
| | 2 | ( 8) | | 2 | ( 31) | | 2 | ( 32) |
| | 3 | ( 8) | | 3 | ( 34) | | 3 | ( 31) |
| | 4 | ( 17) | | 4 | ( 28) | | 4 | ( 33) |
| | 5 | ( 72) | | 5 | ( 45) | | 5 | ( 73) |
| | 6 | (423) | | 6 | (109) | | 6 | ( 721) |
| | 7 | (603) | | 7 | (192) | | 7 | ( 380) |
| | 8 | (201) | | 8 | (154) | | 8 | ( 679) |
| | 9 | ( 22) | | 9 | ( 58) | | 9 | ( 42) |
| | 10 | ( 21) | | 10 | ( 56) | | 10 | ( 32) |
| | 11 | ( 22) | | 11 | ( 34) | | 11 | ( 32) |
| | 12 | ( 34) | | 12 | ( 31) | | 12 | ( 31) |
| 76 | 1 | ( 21) | 79 | 1 | ( 28) | 82 | 1 | ( 32) |
| | 2 | ( 25) | | 2 | ( 32) | | 2 | ( 32) |
| | 3 | ( 25) | | 3 | ( 29) | | 3 | ( 32) |
| | 4 | ( 50) | | 4 | ( 32) | | 4 | ( 32) |
| | 5 | ( 44) | | 5 | ( 66) | | 5 | ( 93) |
| | 6 | (191) | | 6 | (242) | | 6 | (1,070) |
| | 7 | (761) | | 7 | (482) | | 7 | (1,024) |
| | 8 | (306) | | 8 | (384) | | 8 | ( 143) |
| | 9 | ( 33) | | 9 | ( 54) | | 9 | ( 40) |
| | 10 | ( 34) | | 10 | ( 43) | | 10 | ( 40) |
| | 11 | ( 33) | | 11 | ( 32) | | 11 | ( 32) |
| | 12 | ( 40) | | 12 | ( 32) | | 12 | ( 32) |
| 77 | 3 | ( 32) | 80 | 1 | ( 34) | | | |
| | 4 | ( 40) | | 2 | ( 32) | | | |
| | 5 | ( 46) | | 3 | ( 31) | | | |
| | 6 | (489) | | 4 | ( 36) | | | |
| | 7 | (311) | | 5 | ( 62) | | | |
| | 8 | (210) | | 6 | (208) | | | |
| | 9 | ( 42) | | 7 | (238) | | | |
| | 10 | ( 32) | | 8 | (108) | | | |
| | 11 | ( 16) | | 9 | ( 38) | | | |
| | 12 | ( 32) | | 10 | ( 33) | | | |
| | | | | 11 | ( 32) | | | |
| | | | | 12 | ( 32) | | | |

The procedures for establishing this data bank are as follows:




## Reformat Program

The original entrainment larvae fish data were stored in the file ENTXXHISTDEN, where XX indicates the year when the data were collected. The reformat program ELARVARR was then used to convert the entrained larval fish data into a flat-file form and to abbreviate the codes for the species names. These codes are shown in the reformat program (Table 4.38). It is noted that eggs are considered as one category for larvae and are coded as "EG." The computer control statements used are shown below:

```
#RUN *FTN SCARDS=ELARVARR SPUNCH=EL.OBJ
#RUN EL.OBJ 5=ENTXXHISTDEN 6=REFENTXXHDEN
```

## Taxir Create Program

A Taxir Create program ELTAXIRCR was used to create the Entrained.Larvae data bank. The data bank is saved on the COOK tape at position 12. The program ELTAXIRCR is presented in Table 4.39.

TABLE 4.38. Program ELARVARR.

```
C PROGRAM ELARVARR IS USED TO CONVERT THE ENTRAINMENT LARVAE FISH DATA INTO
C A FLAT FILE FORM AND TO ABBREVIATE THE CODES FOR THE SPECIES NAMES.
C UNIT 5 IS THE INPUT FILE.
C UNIT 6 IS THE OUTPUT FILE, (FLAT FILE).
C
C
C
C
C*******
C*******       INITIALIZE VARIABLES.
C*******
C
      INTEGER CASE,CARD,CODE,SNO,MO,DAY
      INTEGER YR,JDAY,MOPD,IGEAR
      INTEGER SERIES,GRATE,NS,IID,DEPTH
      INTEGER STARTH,STARTM,STOPH,STOPM
      INTEGER TTIME,DIEL,REVS,EGGS,C80
      INTEGER CC80,SP,SPDEN(51)
      INTEGER*2 SPEC
      REAL TEMP
      DIMENSION  SPEC(19)
C
C*******
C*******       INITIALIZE DATA.
C*******
C
      DATA SPEC/2HAL,2HSP,2HSM,2HYP,2HTP,2HJD,2HXP,2HSS,2HMS,
     12HCP,2HNS,2HFS,2HQL,2HBR,2HUC,2HXM,2HXC,2HXE,2HXX/
C
C*******
C*******       ICHTHYOPLANKTON SPECIES AND GROUP ENTRAINED AT THE
C*******       D.C. COOK PLANT.
C*******
C
C  ***********************************************************************
C  *  CODE  *  COMMON NAME OR CATEGORY  *   SCIENTIFIC NAME OR CATEGORY   *
C  ***********************************************************************
C  *(1)   AL *  ALEWIFE                  *  ALOSA PSEUDOHARENGUS (WILSON)   *
C  *(2)   SP *  SPOTTAIL SHINER          *  NOTROPIS HUDSONIUS (CLINTON)    *
C  *(3).  SM *  RAINBOW SMELT            *  OSMERUS MORDAX (MITCHILL)       *
C  *(4)   YP *  YELLOW PERCH             *  PERCA FLAVESCENS (MITCHILL)     *
C  *(5)   TP *  TROUT-PERCH              *  PERCOPSIS OMISCOMAYCUS (WALBAUM)*
C  *(6)   JD *  JOHNNY DARTER            *  ETHEOSTOMA NIGRUM (RAFINESQUE)  *
C  *(7)   XP *  UNIDENTIFIED FISH LARVAE *                                  *
C  *         *    AS A RESULT OF POOR    *                                  *
C  *         *    CONDITION              *                                  *
C  *(8)   SS *  SLIMY SCULPIN            *  COTTUS COGNATUS (RICHARDSON)    *
C  *(9)   MS *  MOTTLED SCULPIN          *  COTTUS BAIRDI (GIRARD)          *
C  *(10) CP  *  COMMON CARP              *  CYPRINUS CARPIO (LINNAEUS)      *
C  *(11) NS  *  NINESPINE STICKLEBACK    *  PUNGITIUS PUNGITIUS (LINNAEUS)  *
C  *(12) FS  *  DEEPWATER SCULPIN        *  MYOXOCEPHALUS THOMPSONI (GIRARD)*
C  *(13) QL  *  QUILLBACK                *  CARPIODES CYPRINUS (LESUEUR)    *
C  *(14) BR  *  BURBOT                   *  LOTA LOTA (LINNAEUS)            *
C  *(15) UC  *  UNIDENTIFIED SCULPINS    *  COTTUS SPP.                     *
C  *(16) XM  *  UNIDENTIFIED MINNOWS     *  CYPRINIDAE                      *
C  *(17) XC  *  UNIDENTIFIED COREGONIDS  *  COREGONUS SPP.                  *
C  *(18) XE  *  UNIDENTIFIED DARTERS     *  ETHEOSTOMA SPP.                 *
C  *(19) XX  *  UNIDENTIFIED FISH LARVAE *                                  *
C  *      EG *  EGGS                     *                                  *
C  ***********************************************************************
```

TABLE 4.38. (Continued).

```
C
C*******
C*******        READ THE FIRST DATA LINE FROM THE TRANSREC FILE.
C*******
C
100  READ(5,10,END=999) CASE,CARD,CODE,SNO,MO,DAY,YR,JDAY,
    1MOPD,IGEAR,SERIES,GRATE,NS,IID,DEPTH,STARTH,STARTM,
    1STOPH,STOPM,TTIME,DIEL,TEMP,REVS,EGGS,C80
10   FORMAT(I4,I1,1X,I4,1X,I4,1X,3I2,1X,I3,1X,I2,1X,2I1,1X,2I1,
    11X,I1,I2,1X,2I2,1X,2I2,1X,I4,1X,I1,1X,F4.1,1X,I5,2X,I10,I1)
C
C*******
C*******        WRITE THE FIRST RECORD WHEN THERE ARE NO
C*******        HISTOGRAM VALUES EXISTED.
C*******
C
     WRITE(6,20) CASE,CODE,SNO,MO,DAY,YR,JDAY,MOPD,IGEAR,SERIES,
    1GRATE,NS,IID,DEPTH,STARTH,STARTM,STOPH,STOPM,TTIME,DIEL,
    1TEMP,REVS,EGGS
20   FORMAT(I4,2X,I4,1X,I4,1X,3I2,1X,I3,1X,I2,1X,2I1,1X,2I1,1X,
    1I1,I2,1X,2I2,1X,2I2,1X,I4,1X,I1,1X,F4.1,1X,I5,2X,'EG',2X,I10)
C
     IF(C80.EQ.0) GO TO 100
C
C*******
C*******        READ THE SECOND DATA LINE.
C*******
C
200  READ(5,30,END=999) SP,(SPDEN(L),L=1,51),CC80
30   FORMAT(' ',10X,52I5,5X,I5)
C
C*******
C*******        COMBINE THE FIRST AND SECOND RECORDS WHEN THERE ARE
C*******        MORE THAN ONE HISTOGRAM VALUES.
C*******
C
     DO 40 I=1,51
     IF(SPDEN(I).EQ.0) GO TO 40
     WRITE(6,50)   CASE,CODE,SNO,MO,DAY,YR,JDAY,MOPD,IGEAR,SERIES,
    1GRATE,NS,IID,DEPTH,STARTH,STARTM,STOPH,STOPM,TTIME,DIEL,TEMP,
    1REVS,SPEC(SP),SPDEN(I),I
50   FORMAT(I4,2X,I4,1X,I4,1X,3I2,1X,I3,1X,I2,1X,2I1,1X,2I1,
    11X,I1,I2,1X,2I2,1X,2I2,1X,I4,1X,I1,1X,F4.1,1X,I5,2X,A2,
    12X,I10,2X,I2)
40   CONTINUE
C
     IF(CC80.EQ.0) GO TO 100
     GO TO 200
C
999  STOP
     END
```

TABLE 4.39. Program ELTAXIRCR.

```
RUN *TAXIR
CREATE ENTRAINED.LARVAE.
CASE(FROM O TO 9999).
CODE(FROM O TO 9999).
SAMPLENO(FROM O TO 9999).
MONTH(FROM 1 TO 12).
DAY(FROM 1 TO 31).
YEAR(FROM 75 TO 85).
JDAY(FROM O TO 999).
MONTHPD(FROM O TO 99).
GEAR(FROM O TO 9).
SERIES(FROM O TO 9).
GRATE(FROM O TO 9).
NOR/SOU(FROM O TO 9).
INT/DIS(FROM O TO 9).
DEPTH(FROM O TO 99).
STARTH(FROM O TO 99).
STARTM(FROM O TO 99).
STOPH(FROM O TO 99).
STOPM(FROM O TO 99).
TTIME(FROM O TO 9999).
DIEL(FROM O TO 9).
TEMP(FROM O.O TO 99.9).
REVS(FROM O TO 99999).
SPEC(NAME).
SPDEN(FROM O TO 999999999).
INTERVAL(FROM O TO 99)*
ENTER DATA LOCATION=REFENTHIST. FORMAT=FIXED.
CASE<1-4>.
CODE<7-10>.
SAMPLENO<12-15>.
MONTH<17-18>.
DAY<19-20>.
YEAR<21-22>.
JDAY<24-26>.
MONTHPD<28-29>.
GEAR<31>.
SERIES<32>.
GRATE<34>.
NOR/SOU<35>.
INT/DIS<37>.
DEPTH<38-39>.
STARTH<41-42>.
STARTM<43-44>.
STOPH<46-47>.
STOPM<48-49>.
TTIME<51-54>.
DIEL<56>.
TEMP<58-61>.
REVS<63-67>.
SPEC<70-71>.
SPDEN<74-83>.
INTERVAL<86-87>*
SAVE
STOP
```

NUTRIENT AND ANION

The Nutrients data bank contains 12 descriptors and 10 items. It includes the data from both entrainment and lake samples. The descriptors for this data bank are:

| | |
|---|---|
| 1-STATION | 7-TEMPERATURE |
| 2-MONTH | 8-NITRATE N |
| 3-YEAR | 9-NITRITE N |
| 4-TOTAL P | 10-CHLORIDE |
| 5-ORTHOPHOSPHATE P | 11-SULFATE |
| 6-DISSOLVED SILICA $SiO_2$ | 12-OXYGEN SATURATION |

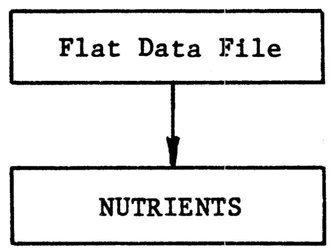
Total P and orthophosphate P are in units of ppb while dissolved silica $SiO_2$, nitrate N, nitrite N, chloride, and sulfate are in units of ppm.

The number of data items stored in the Nutrients data bank is shown in Table 4.40.

TABLE 4.40. The number of data items in the Nutrients data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 74 | 4 | (18) | 78 | 1 | | 81 | 1 | |
| | 5 | (23) | | 2 | | | 2 | |
| | 6 | (18) | | 3 | | | 3 | |
| | 7 | (25) | | 4 | (31) | | 4 | (31) |
| | 8 | (25) | | 5 | | | 5 | |
| | 9 | (11) | | 6 | | | 6 | |
| | 10 | (15) | | 7 | (31) | | 7 | (31) |
| 75 | 4 | (24) | | 8 | | | 8 | |
| | 7 | (30) | | 9 | | | 9 | |
| | 10 | (18) | | 10 | (31) | | 10 | (31) |
| 76 | 1 | | | 11 | | | 11 | |
| | 2 | | | 12 | | | 12 | |
| | 3 | | 79 | 1 | | 82 | 1 | |
| | 4 | (19) | | 2 | | | 2 | |
| | 5 | | | 3 | | | 3 | |
| | 6 | | | 4 | (31) | | 4 | (31) |
| | 7 | (31) | | 5 | | | 5 | |
| | 8 | | | 6 | | | | |
| | 9 | | | 7 | (31) | | | |
| | 10 | | | 8 | | | | |
| | 11 | | | 9 | | | | |
| | 12 | | | 10 | (31) | | | |
| 77 | 3 | | | 11 | | | | |
| | 4 | (31) | | 12 | | | | |
| | 5 | | 80 | 1 | | | | |
| | 6 | | | 2 | | | | |
| | 7 | (31) | | 3 | | | | |
| | 8 | | | 4 | (31) | | | |
| | 9 | | | 5 | | | | |
| | 10 | (31) | | 6 | | | | |
| | 11 | | | 7 | (31) | | | |
| | 12 | | | 8 | | | | |
| | | | | 9 | | | | |
| | | | | 10 | (31) | | | |
| | | | | 11 | | | | |
| | | | | 12 | | | | |

The steps for establishing the Nutrients data bank are shown on the diagram below:




## Taxir Create Program

A Taxir Create program NUTTAXIRCR was used to create the Nutrients data bank which is stored as NUTRIENTS on tape COOK at position 14. The contents of NUTTAXIRCR are shown in Table 4.41.

TABLE 4.41. Program NUTTAXIRCR.

```
R *TAXIR
CREATE NUTRIENTS,
STATION(NAME),
MONTH(FROM 1 TO 12),
YEAR(FROM 74 TO 82),
TOTAL P(FROM 0.000 TO 9999.999),
ORTHOPHOSPHATE P(FROM 0.000 TO 9999.999),
DISSOLVED SILICA SIO2(FROM 0.000 TO 9999.999),
TEMPERATURE C(FROM 0.00 TO 100.00),
NITRATE N(FROM 0.000 TO 9999.999),
NITRITE N(FROM 0.000 TO 9999.999),
CHLORIDE(FROM 0.000 TO 9999.999),
SULFATE(FROM 0.000 TO 9999.999),
OXYGEN SATURATION(FROM 0.000 TO 9999.999)*
ENTER DATA LOCATION=REFREM, FORMAT=FIXED,
STATION<1-20>,
MONTH<21-30>,
YEAR<31-40>,
TOTAL P<41-50>,
ORTHOPHOSPHATE P<51-60>,
DISSOLVED SILICA SIO2<61-70>,
TEMPERATURE C<71-80>,
NITRATE N<81-90>,
NITRITE N<91-100>,
CHLORIDE<101-110>,
SULFATE<111-120>,
OXYGEN SATURATION<121-130>*
SAVE
STOP
```

LAKE WATER CHEMISTRY

The Lakewater data bank contains 48 descriptors and 735 items. The data are solely from lake samples. The descriptors for this data bank are:

1-STATION
2-MONTH
3-YEAR
4-TOTAL PHOSPHORUS-PPB
5-ORTHOPHOSPHORUS-PPB
6-DISSOLVED SILICA-PPM
7-TEMPERATURE-DEGREES C
8-NITRATE-PPM N
9-NITRITE-PPM N
10-CHLORIDE-PPM
11-SULFATE-PPM
12-OXYGEN SATURATION PERCENT
13-ALKALINITY-MEQ/L
14-DISSOLVED BARIUM-PPB
15-DISSOLVED CALCIUM-PPM
16-DISSOLVED CADMIUM-PPM
17-DISSOLVED COBALT-PPB
18-DISSOLVED CHROMIUM-PPB
19-DISSOLVED COPPER-PPB
20-DISSOLVED IRON-PPB
21-DISSOLVED POTASSIUM-PPM
22-DISSOLVED MAGNESIUM-PPM
23-DISSOLVED MANGANESE-PPB
24-DISSOLVED MOLYBDENUM-PPB
25-DISSOLVED SODIUM-PPM
26-DISSOLVED NICKEL-PPB
27-DISSOLVED LEAD-PPB
28-DISSOLVED STRONTIUM-PPB
29-DISSOLVED ZINC-PPB
30-PH
31-SECCHI DISK-M
32-EH-MV
33-CONDUCTIVITY-UMHOS
34-SAMPLE DEPTH-M
35-PARTICULATE BARIUM-PPM
36-PARTICULATE CALCIUM-PPM
37-PARTICULATE COBALT-PPM
38-PARTICULATE CHROMIUM-PPM
39-PARTICULATE COPPER-PPM
40-PARTICULATE IRON-PPM
41-PARTICULATE POTASSIUM-PPM
42-PARTICULATE MAGNESIUM-PPM
43-PARTICULATE MANGANESE-PPM
44-PARTICULATE MOLYBDENUM-PPM
45-PARTICULATE SODIUM-PPM
46-PARTICULATE NICKEL-PPM
47-PARTICULATE STRONTIUM-PPM
48-PARTICULATE ZINC-PPM

The unit for each descriptor is indicated in the descriptors after the sign "-". The number of data items stored in the Lakewater data bank is shown in Table 4.42.

TABLE 4.42. The number of data items in the Lakewater data bank.

| Year | Month | Total # of Items | Year | Month | Total # of Items | Year | Month | Total # of Items |
|---|---|---|---|---|---|---|---|---|
| 74 | 4 | (18) | 76 | 4 | (18) | 80 | 4 | (30) |
| | 5 | (23) | | 7 | (30) | | 7 | (30) |
| | 6 | (18) | 77 | 4 | (30) | | 10 | (30) |
| | 7 | (25) | | 7 | (30) | 81 | 4 | (30) |
| | 8 | (25). | | 10 | (30) | | 7 | (30) |
| | 9 | (11) | 78 | 4 | (30) | | 10 | (30) |
| | 10 | (15) | | 7 | (30) | 82 | 4 | (30) |
| 75 | 4 | (24) | | 10 | (30) | | | |
| | 7 | (24) | 79 | 4 | (30) | | | |
| | 8 | ( 6) | | 7 | (30) | | | |
| | 10 | (18) | | 10 | (30) | | | |

Taxir Create Program

A Taxir Create program TAXSOURCEWAT was used to create the Lakewater data bank which is stored as LAKEWATER on tape COOK at position 15. The content of TAXSOURCEWAT is shown in Table 4.43. The line file from which the Taxir data base was created is named COOKWATER.

TABLE 4.43. Program TAXSOURCENAT.

```
RUN *TAXIR
CREATE LAKEWATER,
STATION(NAME),
MONTH(FROM 1 TO 12),
YEAR(FROM 73 TO 85),
TOTAL PHOSPHORUS-PPB(FROM 0.00 TO 3300.00),
ORTHOPHOSPHORUS-PPB(FROM 0.00 TO 1500.00);
DISSOLVED SILICA-PPM SIO2(FROM 0.00 TO 12.30),
TEMPERATURE-DEGREES C (FROM 0.0 TO 30.0),
NITRATE-PPM N'(FROM 0.00 TO 1.50),
NITRITE-PPM N (FROM 0.000 TO 0.100),
CHLORIDE-PPM (FROM 1.00 TO 110.00),
SULFATE-PPM (FROM 6.00 TO 70.00),
OXYGEN SATURATION PERCENT (FROM 0.0 TO 160.0),
ALKALINITY-MEQ/L (FROM 1.00 TO 5.00),
DISSOLVED BARIUM-PPB (FROM 10.0 TO 130.0),
DISSOLVED CALCIUM-PPM (FROM 20.0 TO 80.0),
DISSOLVED CADMIUM-PPM (FROM 0.120 TO 0.250),
DISSOLVED COBALT-PPB (FROM 0.050 TO 5.000),
DISSOLVED CHROMIUM-PPB (FROM 0.500 TO 3.700),
DISSOLVED COPPER-PPB (FROM 0.900 TO 10.600),
DISSOLVED IRON-PPB (FROM 1.50 TO 460.00),
DISSOLVED POTASSIUM-PPM (FROM 0.700 TO 7.100),
DISSOLVED MAGNESIUM-PPM (FROM 7.40 TO 32.00),
DISSOLVED MANGANESE-PPB (FROM 0.050 TO 184.000),
DISSOLVED MOLYBDENUM-PPB (FROM 2.20 TO 41.50),
DISSOLVED SODIUM-PPM (FROM 2.80 TO 69.50),
DISSOLVED NICKEL-PPB (FROM 2.00 TO 54.50),
DISSOLVED LEAD-PPB (FROM 0.600 TO 0.680),
DISSOLVED STRONTIUM-PPB (FROM 40.0 TO 190.0),
DISSOLVED ZINC-PPB (FROM 0.30 TO 97.00),
PH (FROM 7.40 TO 8.85),
SECCHI DISK-M (FROM 0.65 TO 12.50),
EH-MV (FROM 370 TO 640),
CONDUCTIVITY-UMHOS (FROM 212 TO 672),
SAMPLE DEPTH-M (FROM 0.0 TO 30.0),
PARTICULATE BARIUM-PPM (FROM 0.0000 TO 0.0160),
PARTICULATE CALCIUM-PPM (FROM 0.00 TO 2.32),
PARTICULATE COBALT-PPM (FROM 0.00 TO 0.01),
PARTICULATE CHROMIUM-PPM (FROM 0.0000 TO 0.0110),
PARTICULATE COPPER-PPM (FROM 0.00000 TO 0.00930),
PARTICULATE IRON-PPM (FROM 0.000 TO 1.000),
PARTICULATE POTASSIUM-PPM (FROM 0.000 TO 0.380),
PARTICULATE MAGNESIUM-PPM (FROM 0.000 TO 1.000),
PARTICULATE MANGANESE-PPM (FROM 0.000 TO 0.120),
PARTICULATE MOLYBDENUM-PPM (FROM 0.00000 TO 0.00570),
PARTICULATE SODIUM-PPM (FROM 0.0250 TO 5.8000),
PARTICULATE NICKEL-PPM (FROM 0.00000 TO 0.00270),
PARTICULATE STRONTIUM-PPM (FROM 0.00000 TO 0.00260),
PARTICULATE ZINC-PPM (FROM 0.0000 TO 0.2200)*
ENTER DATA LOCATION=WATER, FORMAT=FIXED,
STATION <1-20>,
MONTH <21-23>,
YEAR <24-26>,
TOTAL PHOSPHORUS-PPB <27-38>,
ORTHOPHOSPHORUS-PPB <39-50>,
DISSOLVED SILICA-PPM SIO2 <51-62>,
TEMPERATURE-DEGREES C <63-74>,
NITRATE-PPM N <75-86>,
NITRITE-PPM N <87-98>,
```

TABLE 4.43. (Continued).

```
CHLORIDE-PPM <99-110>,
SULFATE-PPM <111-122>,
OXYGEN SATURATION PERCENT <123-134>,
ALKALINITY-MEQ/L <135-146>,
DISSOLVED BARIUM-PPB <147-158>,
DISSOLVED CALCIUM-PPM <159-170>,
DISSOLVED CADMIUM-PPM <171-182>,
DISSOLVED COBALT-PPB <183-194>,
DISSOLVED CHROMIUM-PPB <195-206>,
DISSOLVED COPPER-PPB <207-218>,
DISSOLVED IRON-PPB <219-230>,
DISSOLVED POTASSIUM-PPM <231-242>,
DISSOLVED MAGNESIUM-PPM <243-254>,
DISSOLVED MANGANESE-PPB <255-266>,
DISSOLVED MOLYBDENUM-PPB <267-278>,
DISSOLVED SODIUM-PPM <279-290>,
DISSOLVED NICKEL-PPB <291-302>,
DISSOLVED LEAD-PPB <303-314>,
DISSOLVED STRONTIUM-PPB <315-326>,
DISSOLVED ZINC-PPB <327-338>,
PH <339-350>,
SECCHI DISK-M <351-362>,
EH-MV <363-374>,
CONDUCTIVITY-UMHOS <375-386>,
SAMPLE DEPTH-M <387-398>,
PARTICULATE BARIUM-PPM <399-410>,
PARTICULATE CALCIUM-PPM <411-422>,
PARTICULATE COBALT-PPM <423-434>,
PARTICULATE CHROMIUM-PPM <435-446>,
PARTICULATE COPPER-PPM <447-458>,
PARTICULATE IRON-PPM <459-470>,
PARTICULATE POTASSIUM-PPM <471-482>,
PARTICULATE MAGNESIUM-PPM <483-494>,
PARTICULATE MANGANESE-PPM <495-506>,
PARTICULATE MOLYBDENUM-PPM <507-518>,
PARTICULATE SODIUM-PPM <519-530>,
PARTICULATE NICKEL-PPM <531-542>,
PARTICULATE STRONTIUM-PPM <543-554>,
PARTICULATE ZINC-PPM <555-566>*
SAVE
STOP
```

SEDIMENT TEXTURE AND CHEMISTRY

The Sediments data bank contains 52 descriptors and 430 items. The data are from lake samples. The descriptors for this data bank are:

| | |
|---|---|
| 1-STATION | 27-STANDARD DEVIATION OF MEAN GRAIN SIZE-PHI |
| 2-YEAR | 28-KURTOSIS OF GRAIN SIZE |
| 3-STATION DEPTH-M | 29-SKEWNESS OF GRAIN SIZE |
| 4-STATION NUMBER | 30-INSOLUBLE FRACTION-% |
| 5-SAMPLE DEPTH-CM | 31-BARIUM-% |
| 6-LOSS ON IGNITION-% | 32-TOTAL CARBON-% |
| 7-WATER CONTENT-% | 33-INORGANIC CARBON--% |
| 8--3 PHI-% | 34-ORGANIC CARBON-% |
| 9--2 PHI-% | 35-CALCIUM-% |
| 10--1 PHI-% | 36-COBALT-% |
| 11-0 PHI-% | 37-CHROMIUM-% |
| 12-1 PHI-% | 38-COPPER-% |
| 13-2 PHI-% | 39-IRON-% |
| 14-3 PHI-% | 40-POTASSIUM-% |
| 15-4 PHI-% | 41-MAGNESIUM-% |
| 16-5 PHI-% | 42-MANGANESE-% |
| 17-6 PHI-% | 43-MOLYBDENUM-% |
| 18-7 PHI-% | 44-SODIUM-% |
| 19-8 PHI-% | 45-NICKEL-% |
| 20-9 PHI-% | 46-LEAD-% |
| 21-10 PHI-% | 47-STRONTIUM-% |
| 22-GRAVEL-% | 48-ZINC--% |
| 23-SAND-% | 49-EH-MV |
| 24-SILT-% | 50-PH |
| 25-CLAY-% | 51-X-LOCATION |
| 26-MEAN GRAIN SIZE-PHI | 52-Y-LOCATION |

The number of data items stored in the Sediments data bank is shown in Table 4.44.

TABLE 4.44. The number of data items in the Sediments data bank.

| Year | Total Number of Items |
| --- | --- |
| 1973 | (158) |
| 1975 | (160) |
| 1977 | (112) |

Taxir Create Program

A Taxir Create program TAXSOURCESED was used to create the Sediments data bank which is stored as SEDIMENTS on tape COOK at position 16. The content of TAXSOURCESED is shown in Table 4.45. The line file from which the Taxir data base was created is named COOKSEDIMENT.

TABLE 4.45. Program TAXSOURCESED.

```
RUN *TAXIR
CREATE SEDIMENTS,
STATION(NAME),
YEAR(FROM 1973 TO 1977),
STATION DEPTH-M (FROM 4.5 TO 61.5),
STATION NUMBER (FROM 1. TO 233.),
SAMPLE DEPTH-CM (FROM 0.5 TO 62.5),
LOSS ON IGNITION-% (FROM 0.7 TO 60.4),
WATER CONTENT-% (FROM 14.4 TO 95.3),
-3 PHI-% (FROM 0.00 TO 8.47),
-2 PHI-% (FROM 0.00 TO 20.71),
-1 PHI-% (FROM 0.00 TO 42.38),
0 PHI-% (FROM 0.00 TO 57.15),
1 PHI-% (FROM 0.00 TO 65.41),
2 PHI-% (FROM 0.00 TO 82.51),
3 PHI-% (FROM 0.00 TO 81.01),
4 PHI-% (FROM 0.00 TO 85.70),
5 PHI-% (FROM 0.00 TO 39.40),
6 PHI-% (FROM 0.00 TO 26.86),
7 PHI-% (FROM 0.00 TO 64.76),
8 PHI-% (FROM 0.00 TO 23.58),
9 PHI-% (FROM 0.00 TO 9.36),
10 PHI-% (FROM 0.00 TO 19.62),
GRAVEL-% (FROM 0.00 TO 63.10),
SAND-% (FROM 15.60 TO 102.00),
SILT-% (FROM 0.00 TO 62.30),
CLAY-% (FROM 0.00 TO 47.90),
MEAN GRAIN SIZE-PHI (FROM -1.300 TO 6.100),
STANDARD DEVIATION OF MEAN GRAIN SIZE-PHI (FROM 0.130 TO 2.900),
KURTOSIS OF GRAIN SIZE (FROM -5.600 TO 2.500),
SKEWNESS OF GRAIN SIZE (FROM -3.000 TO 35.000),
INSOLUBLE FRACTION-% (FROM 6.00 TO 99.00),
BARIUM-% (FROM 0.000000 TO 0.019000),
TOTAL CARBON-% (FROM 0.15 TO 7.45),
INORGANIC CARBON-% (FROM 0.00 TO 5.30),
ORGANIC CARBON-% (FROM 0.00 TO 5.30),
CALCIUM-% (FROM 0.000 TO 11.700),
COBALT-% (FROM 0.000140 TO 0.004200),
CHROMIUM-% (FROM 0.000018 TO 0.025000),
COPPER-% (FROM 0.000020 TO 0.005000),
IRON-% (FROM 0.120 TO 9.200),
POTASSIUM-% (FROM 0.035 TO 0.760),
MAGNESIUM-% (FROM 0.000 TO 8.740),
MANGANESE-% (FROM 0.000730 TO 0.095000),
MOLYBDENUM-% (FROM 0.000000 TO 0.003580),
SODIUM-% (FROM 0.0100 TO 0.0860),
NICKEL-% (FROM 0.000000 TO 0.010300),
LEAD-% (FROM 0.000094 TO 0.001090),
STRONTIUM-% (FROM 0.000380 TO 0.010700),
ZINC-% (FROM 0.00100 TO 0.03000),
EH-MV (FROM -70.0 TO 485.0),
PH (FROM 6.95 TO 8.55),
X-LOCATION (FROM 1.00 TO 18.00),
Y-LOCATION (FROM 0.75 TO 10.00)*
ENTER DATA LOCATION=COOKSEDIMENT, FORMAT=FIXED,
STATION <1-16>,
YEAR <17-21>,
STATION DEPTH-M <22-31>,
STATION NUMBER <32-41>,
SAMPLE DEPTH-CM <42-51>,
```

TABLE 4.45. (Continued).

```
LOSS ON IGNITION-% <52-61>,
WATER CONTENT-% <62-71>,
-3 PHI-% <72-81>,
-2 PHI-% <82-91>,
-1 PHI-% <92-101>,
0 PHI-% <102-111>,
1 PHI-% <112-121>,
2 PHI-% <122-131>,
3 PHI-% <132-141>,
4 PHI-% <142-151>,
5 PHI-% <152-161>,
6 PHI-% <162-171>,
7 PHI-% <172-181>,
8 PHI-% <182-191>,
9 PHI-% <192-201>,
10 PHI-% <202-211>,
GRAVEL-% <212-221>,
SAND-% <222-231>,
SILT-% <232-241>,
CLAY-% <242-251>,
MEAN GRAIN SIZE-PHI <252-261>,
STANDARD DEVIATION OF MEAN GRAIN SIZE-PHI <262-271>,
KURTOSIS OF GRAIN SIZE <272-281>,
SKEWNESS OF GRAIN SIZE <282-291>,
INSOLUBLE FRACTION-% <292-301>,
BARIUM-% <302-311>,
TOTAL CARBON-% <312-321>,
INORGANIC CARBON-% <322-331>,
ORGANIC CARBON-% <332-341>,
CALCIUM-% <342-351>,
COBALT-% <352-361>,
CHROMIUM-% <362-371>,
COPPER-% <372-381>,
IRON-% <382-391>,
POTASSIUM-% <392-401>,
MAGNESIUM-% <402-411>,
MANGANESE-% <412-421>,
MOLYBDENUM-% <422-431>,
SODIUM-% <432-441>,
NICKEL-% <442-451>,
LEAD-% <452-461>,
STRONTIUM-% <462-471>,
ZINC-% <472-481>,
EH-MV <482-491>,
PH <492-501>,
X-LOCATION <502-511>,
Y-LOCATION <512-521>*
SAVE
STOP
```

# CHAPTER 5

# INTERACTIVE PROGRAM

## PROGRAM INTERACT

The program INTERACT (Table 5.1) is an interactive computer program which is written to provide on-line instructions to users wishing to access the Cook water quality data base. It is hoped that this interactive program can facilitate the use of this data base by users with little or no knowledge of computer programming, MTS, or Taxir. The program INTERACT consists of seven major commands for accessing the data base. A brief description of these commands is given below.

1. "H" = HELP

   This program offers a HELP file (Table 5.2) which users can request at any point in the data accessing process. By simply typing in the command "H," users are provided with a complete explanation of other commands used in connection with the data base.

2. "S" = SELECT DATA BANK

   Any one of the 13 data banks contained in the data base system can be selected using this command. Since it is impossible to keep all 13 data banks on line at all times, an additional program has been written to access those data which are stored on a computer tape. This FORTRAN program is called LINK (Table 5.3). It will select and restore information from a tape whenever a file restoration request is made. When the job is completed, the computer will return to this main program INTERACT.

3. "O" = SELECT OUTPUT OPTION

   This command provides three possible options for output: terminal screen, temporary or permanent line file, and MTS line or page printer.

4. "V" = VARIABLE DESCRIPTION

   The list and types of descriptors in this data base management system can be obtained using this command.

5. "P" = PERCENTAGE OF TOTAL DATA ITEMS

   The percentage of total data items can be obtained by applying this command.

6. "Q" = QUERY

   This is an important command for retrieving data. By typing "Q," users can query the needed information and generate necessary tables and reports with requested data.

7. "T" = TERMINATE THE PROGRAM

   Entering "T" results in the termination of the execution of the program.

In the following presentation of the computer programs INTERACT and LINK, boxes have been used to provide information and explanations of the different stages of operations, as well as the assignments for the input and output units and devices.

## INTERFACE WITH TAXIR USING INTERACT

INTERACT is a FORTRAN-based interactive program. It acts as an intermediary between the users and Taxir by asking the users a series of questions

TABLE 5.1. Program INTERACT.

```
C THIS  INTERACTIVE  PROGRAM INTERFACES WITH TAXIR TO  RETRIEVE DATA
C FROM THE  13  DATA BANKS  CREATED FOR COOK PROJECT. TO WORK WITH A
C PARTICULAR  DATA BANK,  THE DATA SHOULD BE  AVAILABLE ON LINE IN A
C TEMPORARY OR  PERMANENT FILE.  HOWEVER, THIS PROGRAM HAS AN OPTION
C TO RESTORE THE DATA BANKS FROM A TAPE.
C
C TO RUN THIS PROGRAM TYPE: "SOURCE INTERACTIVE"
C
C IN THE RUN COMMAND UNITS ARE ASSIGNED AS FOLLOW:
C UNIT 5 IS ASSIGNED TO *MSOURCE* TO READ THE INPUT FROM THE SCREEN.
C UNIT 6 IS ASSIGNED TO *SINK* TO PRINT THE  MESSAGES ON THE SCREEN.
C TAXIR  MESSAGES ARE  WRITTEN ON SERCOM. IN THIS  PROGRAM SERCOM IS
C ASSIGNED TO A TEMPORARY FILE CALLED "-CHECK".
C UNIT 7 IS ALSO ASSIGNED TO "-CHECK" TO CHECK TAXIR MESSAGES.
C
C
C
C      **************************************************
C      *                                                *
C      *        MAIN PROGRAM TO INITIALIZE & CALL       *
C      *        SYSTEM SUBROUTINES.                     *
C      *                                                *
C      **************************************************
C
C
C
      LOGICAL*1   RESP
      LOGICAL     EQUC
C
C
      CALL  INITLZ
      WRITE(6,1)
1     FORMAT(//,'ARE YOU ALREADY FAMILIAR WITH THE PROGRAM?'/,
     1'TYPE "Y" FOR YES TO SKIP THE DESCRIPTION OF THE COMMANDS.'/,
     1'TYPE "N" IF YOU NEED HELP.')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 1000
      CALL HELP(0)
C
C
1000  CALL CMDNOE('$EM -CHECK',10)
      WRITE(6,4)
4     FORMAT(//,'SELECT NEXT COMMAND: (H,S,O,V,P,Q,T)')
C
C*******
C*******        ACCEPT A COMMAND AND DECODE IT.
C*******
C
      CALL DECODE(NCOMND)
      GO TO (10,20,30,40,50,60,70), NCOMND
C
10    CALL HELP(1)
      GO TO 1000
C
20    CALL SELDB
      GO TO 1000
C
30    CALL OUTOP
```

TABLE 5.1. (Continued).

```
      GO TO 1000
C
40    CALL VOCAB
      GO TO 1000
C
50    CALL PERCNT
      GO TO 1000
C
60    CALL QUERY
      GO TO 1000
C
70    CALL TERMIN
      GO TO 1000
C
C
      END
      SUBROUTINE INITLZ
C
C
C
C       ****************************************************************
C       *                                                              *
C       *         SUBROUTINE INITLZ; TO INITIALIZE TAXIR PROGRAM.     *
C       *                                                              *
C       ****************************************************************
C
C
C
      CALL TAXIR('DUMMY',0)
      CALL FREAD(-2,'LENGTH',.TRUE.)
      CALL FREAD(-2,'DELIMITERS','/%¢/;/')
C
C
      RETURN
      END
      SUBROUTINE  DECODE(NCOMND)
C
C
C
C       ***********************************************************
C       *                                                         *
C       *         SUBROUTINE DECODE; TO RECEIVE THE COMMAND &     *
C       *         DECODE IT.                                      *
C       *                                                         *
C       ***********************************************************
C
C
C
      LOGICAL*1 RESP
      LOGICAL EQUC
C
C
      NERR=0
10    NCOMND=0
      LEN=1
      CALL FREAD(5,'STRING:',RESP,LEN)
      IF(EQUC('H',RESP).OR.EQUC('h',RESP)) NCOMND=1
      IF(EQUC('S',RESP).OR.EQUC('s',RESP)) NCOMND=2
      IF(EQUC('O',RESP).OR.EQUC('o',RESP)) NCOMND=3
      IF(EQUC('V',RESP).OR.EQUC('v',RESP)) NCOMND=4
```

TABLE 5.1. (Continued).

```
      IF(EQUC('P',RESP).OR.EQUC('p',RESP)) NCOMND=5
      IF(EQUC('Q',RESP).OR.EQUC('q',RESP)) NCOMND=6
      IF(EQUC('T',RESP).OR.EQUC('t',RESP)) NCOMND=7
C
C
      IF(NCOMND.GT.0) RETURN
C
C
      NERR=NERR+1
      IF(NERR.GE.2) GO TO 20
C
      WRITE(6,1)
1     FORMAT(//,'INVALID COMMAND, TRY AGAIN.')
      GO TO 10
C
C*******
C*******        PRINT THE LIST OF COMMANDS.
C*******
C
20    WRITE(6,2)
2     FORMAT(//,'INVALID COMMAND! THE COMMANDS ARE:'//,
     1'H     HELP'/,
     1'S     SELECT DATA BANK'/,
     1'O     SELECT OUTPUT OPTION'/,
     1'V     VARIABLE DESCRIPTION'/,
     1'P     PERCENTAGE OF TOTAL DATA ITEMS'/,
     1'Q     QUERY'/,
     1'T     TERMINATE THE PROGRAM!')
C
C
      NERR=0
      GO TO 10
C
C
      END
      SUBROUTINE  HELP(I)
C
C
C
C      **************************************************************
C      *                                                            *
C      *        SUBROUTINE HELP ; THIS ROUTINE IS USED TO HELP      *
C      *        THE USER TO UNDERSTAND THE QUERY COMMANDS USED      *
C      *        IN THE PROGRAM. THE PROGRAM CONTENTS ARE READ       *
C      *        FROM THE MTS LINE FILE "SDLS:HELP".                 *
C      *                                                            *
C      **************************************************************
C
C
C
      IF(I.NE.0) GO TO 5
      CALL CMDNOE('$COPY SDLS:HELP(001,071) TO *SINK*@SP ',38)
      RETURN
C
C
5     WRITE(6,1)
1     FORMAT(//,'HELP: SELECT THE COMMAND THAT YOU WANT TO '/,
     1'BE EXPLAINED: (H,S,O,V,P,Q,T)')
C
C
```

TABLE 5.1. (Continued).

```
      CALL DECODE(NCOMND)
C
      GO TO(10,20,30,40,50,60,70), NCOMND
C
10    CALL CMDNOE('$COPY SDLS:HELP(072,089) TO *SINK*@SP ',38)
      RETURN
C
20    CALL CMDNOE('$COPY SDLS:HELP(090,104) TO *SINK*@SP ',38)
      RETURN
C
30    CALL CMDNOE('$COPY SDLS:HELP(105,120) TO *SINK*@SP ',38)
      RETURN
C
40    CALL CMDNOE('$COPY SDLS:HELP(121,135) TO *SINK*@SP ',38)
      RETURN
C
50    CALL CMDNOE('$COPY SDLS:HELP(136,151) TO *SINK*@SP ',38)
      RETURN
C
60    CALL CMDNOE('$COPY SDLS:HELP(152,166) TO *SINK*@SP ',38)
      RETURN
C
70    CALL CMDNOE('$COPY SDLS:HELP(167,181) TO *SINK*@SP ',38)
C
C
      RETURN
      END
      SUBROUTINE SELDB
C
C
C
C       *******************************************************
C       *                                                     *
C       *       SUBROUTINE SELDB; TO SELECT A SPECIFIC        *
C       *       DATA BANK.                                    *
C       *                                                     *
C       *******************************************************
C
C
C
      INTEGER ERRNUM
      LOGICAL*1  RESP
      LOGICAL    EQUC
C
C*******
C*******        PRINT THE LIST OF DATA BANKS.
C*******
C
10    WRITE(6,2)
2     FORMAT(//,'THE DATA BANKS AND THEIR CODE NUMBERS ARE;',
     1//,
     1'LAKE.PHYTOPLANKTON                 OR    1'/,
     1'ENTRAINED.PHYTOPLANKTON            OR    2'/,
     1'LAKE.ZOOPLANKTON                   OR    3'/,
     1'ENTRAINED.ZOOPLANKTON              OR    4'/,
     1'LAKE.BENTHOS                       OR    5'/,
     1'ENTRAINED.BENTHOS                  OR    6'/,
     1'IMPINGED.BENTHOS                   OR    7'/,
     1'ADULT.FISH.SUMMARY.STATISTICS      OR    8'/,
     1'LAKE.ADULT.FISH                    OR    9'/,
```

TABLE 5.1. (Continued).

```
     1'IMPINGED.ADULT.FISH                  OR  10'/,
     1'LAKE.LARVAE                          OR  11'/,
     1'ENTRAINED.LARVAE                     OR  12'/,
     1'NUTRIENT.AND.ANION                   OR  13'/,
     1'LAKEWATER.CHEMISTRY                  OR  14'/,
     1'SEDIMENTS                            OR  15')
C
C*******
C*******       CHECK THE AVAILABILITY OF THE DATA BANK(S).
C*******
C
20    WRITE(6,3)
3     FORMAT(//,'THE DATA SHOULD BE AVAILABLE ON LINE.'/,
     1'IS THE DATA BANK OF YOUR INTEREST AVAILABLE ON LINE?'/,
     1'PLEASE ANSWER "Y" OR "N".')
C
C
      LEN=1
      CALL FREAD(5,'STRING:',RESP,LEN)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 30
      CALL CMDNOE('SOURCE RESTORE',14)
      STOP
C
C
30    WRITE(6,4)
4     FORMAT(//,'TO PLACE THE REQUESTED DATA BANK IN THE TAXIR PROGRAM,'/,
     1'ENTER THE DATA BANK CODE NUMBER:'/,
     1'(EX: ENTER 1 FOR LAKE.PHYTOPLANKTON)')
C
C
      CALL FREAD(5,'I:',IC)
      GO TO (110,120,130,140,150,160,170,180,190,200,210,220,230,240,250),IC
C
C
      WRITE(6,5)
5     FORMAT(//,'BAD OPTION!')
      GO TO 10
C
C*******
C*******       TAXIR INTERFACE WITH THE DATA BASE.
C*******
C
110   CALL TAXIR('GET LAKE.PHYTOPLANKTON',22)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -LAKE.PHYTO',15)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
120   CALL TAXIR('GET ENTRAINED.PHYTOPLANKTON',27)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -ENT.PHYTO',14)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
```

TABLE 5.1. (Continued).

```
      GO TO 300
C
130   CALL TAXIR('GET LAKE.ZOOPLANKTON',20)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -LAKE.ZOO',13)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
140   CALL TAXIR('GET ENTRAINED.ZOOPLANKTON',25)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -ENT.ZOO',12)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
150   CALL TAXIR('GET LAKE.BENTHOS',16)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -LAKE.BEN',13)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
160   CALL TAXIR('GET ENTRAINED.BENTHOS',21)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -ENT.BEN',12)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
170   CALL TAXIR('GET IMPINGED.BENTHOS',20)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -IMP.BEN',12)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
180   CALL TAXIR('GET ADULT.FISH.SUMMARY.STATISTICS',33)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -AD.FISH.S.S',16)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
```

TABLE 5.1. (Continued).

```
      GO TO 300
C
190   CALL TAXIR('GET LAKE.ADULT.FISH',19)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -L.AD.FISH',14)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
200   CALL TAXIR('GET IMPINGED.ADULT.FISH',23)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -I.AD.FISH',14)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
210   CALL TAXIR('GET LAKE.LARVAE',15)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -L.LARVAE',13)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
220   CALL TAXIR('GET ENTRAINED.LARVAE',20)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -E.LARVAE',13)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
C
230   CALL TAXIR('GET NUTRIENTS',13)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -NUTRIENTS',14)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
240   CALL TAXIR('GET LAKEWATER',13)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      CALL TAXIR('GET -LAKEWATER',14)
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) RETURN
      GO TO 300
```

TABLE 5.1. (Continued).

```
250  CALL TAXIR('GET SEDIMENTS',12)
     CALL ERRCHK(ERRNUM)
     IF(ERRNUM.EQ.0) RETURN
     CALL TAXIR('DUMMY',-1)
     CALL CMDNOE('$EM -CHECK',10)
     CALL TAXIR('GET -SEDIMENTS',14)
     CALL ERRCHK(ERRNUM)
     IF(ERRNUM.EQ.0) RETURN
C
C*******
C*******        CHECK TAXIR MESSAGES.
C*******
C
300  CALL TAXIR('DUMMY',-1)
     CALL CMDNOE('$EM -CHECK',10)
     WRITE(6,6)
6    FORMAT(//,'REQUESTED DATA BANK IS NOT AVAILABLE.'/,
    1'DO YOU WANT TO SELECT ANOTHER DATA BANK?'/,
    1'ENTER "Y" FOR YES, "N" FOR NO.')
     CALL FREAD(5,'STRING:',RESP,LEN)
     IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 20
C
C
     RETURN
     END
     SUBROUTINE OUTOP
C
C
C
C      *****************************************************
C      *                                                   *
C      *       SUBROUTINE OUTOP; TO SELECT AN OUTPUT       *
C      *       FILE OR DEVICE.                             *
C      *                                                   *
C      *****************************************************
C
C
C
     LOGICAL*1  LINE(29)
C
C
     DATA LINE(1),LINE(2),LINE(3),LINE(4),LINE(5),LINE(6),
    1LINE(7)/1HS,1HE,1HT,1H ,1HO,1HU,1HT/
     DATA LINE(8),LINE(9),LINE(10),LINE(11)/1HP,1HU,1HT,1H=/
C
C*******
C*******        OUTPUT FILE/DEVICE OPTIONS.
C*******
C
5    WRITE(6,1)
1    FORMAT(//,'SELECT OUTPUT FILE/DEVICE, TYPE:'//,
    13X,'1  TERMINAL'/,
    13X,'2  LINE PRINTER'/,
    13X,'3  OUTPUT FILE NAME')
C
C
     CALL FREAD(5,'I:',IC)
     IF(IC.EQ.1) GO TO 10
     IF(IC.EQ.2) GO TO 20
     IF(IC.EQ.3) GO TO 30
```

```
C
C
      WRITE(6,2)
2     FORMAT(//,'BAD OPTION!')
      GO TO 5
C
C*******
C*******       INTERFACE WITH TAXIR.
C*******
C
10    CALL TAXIR('SET OUTPUT=*MSINK*',18)
      RETURN
C
20    CALL TAXIR('SET OUTPUT=*PRINT*',18)
      RETURN
C
30    WRITE(6,3)
3     FORMAT(//,'ENTER NAME OF THE OUTPUT FILE.')
C
C*******
C*******       INTERFACE WITH TAXIR.
C*******
C
      LEN=18
      CALL FREAD(5,'STRING:',LINE(12),LEN)
      LEN=LEN+11
      CALL TAXIR(LINE,LEN)
C
C
      RETURN
      END
      SUBROUTINE VOCAB
C
C
C
C       ****************************************************************
C       *                                                              *
C       *       SUBROUTINE VOCAB;  TO PROVIDE THE USER WITH THE        *
C       *       LIST OF DESCRIPTORS FOR THE SELECTED DATA BANK.        *
C       *                                                              *
C       ****************************************************************
C
C
C
      INTEGER ERRNUM
      LOGICAL*1 LINE(87),ASTRIS,RESP
      LOGICAL   EQUC
C
C
      DATA ASTRIS/1H*/
      DATA LINE(1),LINE(2),LINE(3),LINE(4)/1HS,1HH,1HO,1HW/
      DATA LINE(5)/1H /
C
C*******
C*******       OBTAIN DICTIONARY INFORMATION.
C*******
C
      WRITE(6,1)
1     FORMAT(//,'ARE YOU FAMILIAR WITH ALL THE OPTIONS FOR OBTAINING'/,
     1'INFORMATION ABOUT DATA DESCRIPTORS?'/,
```

TABLE 5.1. (Continued).

```
    1'TYPE "Y" FOR YES TO SKIP THE DESCRIPTIONS.'/,
    1'TYPE "N" IF YOU NEED THE DESCRIPTIONS.')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 10
C
C
40    WRITE(6,2)
2     FORMAT(//,'THE OPTIONS ARE LISTED BELOW:'///,
     1'OPTION 1:  TYPE "SUM" TO GET THE DATA SUMMARY FOR THE'/,
     1'           SELECTED DATA BANK.')
      WRITE(6,3)
3     FORMAT(//,'OPTION 2:  TO OBTAIN INFORMATION ON DESCRIPTORS,',
     1' TYPE'/,
     1'           THE NAMES  OR  CODES  OF THE  DESCRIPTORS,'/,
     1'           (SEPARATED BY ,).'/,
     1'           ENTER "ALL" FOR ALL THE DESCRIPTORS.'/,
     1'           (EX: 1,2,3 OR MONTH,DAY OR ALL)')
      WRITE(6,4)
4     FORMAT(//,'OPTION 3:  FOR LISTINGS OF BOTH DESCRIPTORS AND',
     1' THEIR'/,
     1'           STATES, TYPE  "F"  FOLLOWING  A  PARAMETER'/,
     1'           NAME OR CODE AS IN THE EXAMPLE BELOW.'/,
     1'           (EX: 1 F,2 F,3 F OR MONTH F,DAY F OR ALL F)')
      WRITE(6,5)
5     FORMAT(//,'OPTION 4:  TO OBTAIN THE LIST OF THE CODE(S) AND',
     1' NAME'/,
     1'           (S)  OF DESCRIPTOR  STATE(S) THAT CONTAINS'/,
     1'           OR BEGINS WITH A PARTICULAR  STRING;  TYPE'/,
     1'           THE  DESCRIPTOR  NAME OR CODE  ALONG  WITH'/,
     1'           THE WORDS  "BEGINS" OR "CONTAINS"  AND THE'/,
     1'           SPECIFIED STRING.'/,
     1'           (EX: NAME CONTAINS MARY OR NAME BEGINS M)'/,
     1'          *NOTE  THAT FOR THIS OPTION THE  DESCRIPTOR'/,
     1'           TYPE SHOULD BE NAME OR ORDER.')
C
C
10    WRITE(6,6)
6     FORMAT(//,'ENTER THE DESCRIPTOR NAME(S) OR CODE(S) ALONG WITH'/,
     1'THE REQUIRED STRING (IF ANY).')
C
C*******
C*******       TAXIR INTERFACE WITH THE DATA BASE.
C*******
C
      LEN=80
      CALL FREAD(5,'STRING:',LINE(6),LEN)
      LEN=LEN+6
      LINE(LEN)=ASTRIS
C
C
      CALL TAXIR(LINE,LEN)
C
C*******
C*******       CHECK TAXIR MESSAGES.
C*******
C
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) GO TO 20
```

TABLE 5.1. (Continued).

```
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      WRITE(6,7)
7     FORMAT(//,'THE INFORMATION YOU ASKED CAN NOT BE LISTED.'/,
     1'THE STATEMENT YOU HAVE TYPED CONTAINS ERROR(S).'/,
     1'DO YOU WANT TO TRY AGAIN? (Y/N)')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 30
      GO TO 20
C
C
30    WRITE(6,8)
8     FORMAT(//,'DO YOU NEED HELP TO CONSTRUCT YOUR QUESTION? (Y/N)!)
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 40
      GO TO 10
C
C
20    RETURN
      END
      SUBROUTINE PERCNT
C
C
C
C       ************************************************************
C       *                                                          *
C       *         SUBROUTINE PERCNT; TO PRODUCE STATISTICS IN      *
C       *         TERMS OF PERCENTAGES FOR A DATA BANK.            *
C       *                                                          *
C       ************************************************************
C
C
C
      INTEGER ERRNUM
      LOGICAL*1  LINE(86),ASTRIS,RESP
      LOGICAL EQUC
C
C
      DATA LINE(1),LINE(2),LINE(3),LINE(4),LINE(5)/1HH,1HM,1H ,1H*,1H /
      DATA ASTRIS/1H*/
C
C*******
C*******       INSTRUCTIONS TO OBTAIN THE NECESSARY STATISTICS.
C*******
C
30    WRITE(6,1)
1     FORMAT(//,'DO YOU NEED HELP TO CONSTRUCT YOUR QUESTION? (Y/N)')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 10
      GO TO 20
C
C
10    WRITE(6,2)
2     FORMAT(//,'TO SPECIFY YOUR QUESTION YOU SHOULD ENTER THE',
     1' NAME'/, 'OR CODE OF THE  DESCRIPTOR  ALONG  WITH ITS  TYPE,'/,
     1'(SEPARATED BY ,).'/,
```

TABLE 5.1. (Continued).

```
     1'(EX: DAY,31 OR YEAR,80)'//,
     1'TO COMBINE THE DESCRIPTORS USE LOGICAL  STATEMENTS'/,
     1'"AND/OR". YOU CAN ALSO USE PARENTHESES TO NAME THE'/,
     1'SUBSET OF INTEREST FROM THE TOTAL BANK.'/,
     1'EX: YEAR,80 AND MONTH,MAY'/,
     1'     YEAR,80 OR  MONTH,MAY'/,
     1'     (YEAR,80 AND MONTH,MAY) OR DAY,31')
C
C
20    WRITE(6,3)
3     FORMAT(//,'ENTER THE QUESTION.')
C
C*******
C*******        TAXIR INTERFACE WITH THE DATA BASE.
C*******
C
      LEN=80
      CALL FREAD(5,'STRING:',LINE(6),LEN)
      LEN=LEN+6
      LINE(LEN)=ASTRIS
C
C
      CALL TAXIR(LINE,LEN)
C
C*******
C*******        CHECK TAXIR MESSAGES.
C*******
C
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) GO TO 40
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      WRITE(6,4)
4     FORMAT(//,'THE PERCENTAGE CAN NOT BE COMPUTED.'/,
     1'THE STATEMENT YOU HAVE TYPED CONTAINS ERROR(S).'/,
     1'DO YOU WANT TO TRY AGAIN? (Y/N)')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 30
C
C
40    RETURN
      END
      SUBROUTINE QUERY
C
C
C
C        *******************************************************
C        *                                                     *
C        *      SUBROUTINE QUERY; TO RETRIEVE INFORMATION      *
C        *      FROM THE SELECTED DATA BANK.                   *
C        *                                                     *
C        *******************************************************
C
C
C
      INTEGER ERRNUM
      LOGICAL*1  LINE(166),ASTRIS,RESP
      LOGICAL EQUC
```

TABLE 5.1. (Continued).

```
C
C
      DATA  ASTRIS/1H*/
      DATA  LINE(1),LINE(2)/1HQ,1H /
C
C*******
C*******        INSTRUCTIONS TO CONSTRUCT THE QUERIES.
C*******
C
40    WRITE(6,1)
1     FORMAT(//,'DO YOU NEED THE INSTRUCTIONS TO MAKE YOUR QUERY? (Y/N)')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 10
      GO TO 20
C
C
10    WRITE(6,2)
2     FORMAT(//,'THE QUERY "Q" COMMAND IS THE MOST POWERFUL RETRIEVAL',
     1' TOOL'/,'IN THIS PROGRAM.  THE  CONSTRUCTION OF A  QUERY',
     1' STATEMENT'/,'CONSISTS OF TWO PARTS:'///,
     1'I.  THE SPECIFICATION OF THE  DESCRIPTOR(S) WHICH MUST BE'/,
     1'      LISTED IN THE DATA OUTPUT. THERE ARE FIVE OPTIONS FOR'/,
     1'      THIS PART AS SHOWN BELOW:'//,
     1'      1.  THE NAME(S)  OR  CODE(S) OF THE DESCRIPTOR(S) CAN'/,
     1'          BE ENCLOSED IN PARENTHESES, (SEPARATED BY ,).'/,
     1'          TYPING  AN  "A"  IN ENCLOSED ANGLE BRACKETS  "<>"'/,
     1'          FOLLOWING DESCRIPTOR CODE OR NAME WILL  RESULT IN'/,
     1'          A PRINT OF DUPLICATE STATES.'/,
     1'          EX: (YEAR,MONTH,DAY)'/,
     1'              (YEAR<A>,MONTH<A>,DAY<A>)'//,
     1'      2.  TYPING  "ALL"  RESULTS IN THE LIST FOR ALL OF THE'/,
     1'          DESCRIPTORS.'/,
     1'          EX: (ALL)')
     WRITE(6,3)
3    FORMAT(/,'      3.  A TOTAL OR SUBTOTAL OF NUMERICAL  DESCRIPTORS CAN'/,
     1'          BE OBTAINED BY TYPING A  "TN"  IN A PAIR OF ANGLE'/,
     1'          BRACKETS,  WHERE N INDICATES THE SUBTOTAL FOR THE'/,
     1'          NTH  DESCRIPTOR.  IF N  IS  ZERO,  THEN THE GRAND'/,
     1'          TOTAL IS PRINTED.  AN  EXAMPLE  IS SHOWN BELOW IN'/,
     1'          WHICH TO IS THE GRAND TOTAL AND T1 AND T2 ARE THE'/,
     1'          SUBTOTALS FOR DESCRIPTORS 1 AND 2.'/,
     1'          EX: (YEAR,MONTH,SPCONT<T0,T1,T2>)'//,
     1'      4.  IF THE OUTPUT IS DESIGNED TO SERVE AS AN INPUT TO'/,
     1'          MIDAS  FOR   FURTHER   STATISTICAL  ANALYSIS  THE'/,
     1'          STATEMENT  "<STAT,FN>" SHOULD BE TYPED BEFORE THE'/,
     1'          DESCRIPTOR LIST,  WHERE  "FN"  IS THE NAME OF THE'/,
     1'          OUTPUT FILE PROVIDED BY THE "O" COMMAND.'/,
     1'          EX: <STAT,RESULT> (YEAR,MONTH,DAY,SPCONT)'//,
     1'      5.  IF THE  WORD  "TAB"  IS  USED IN  A PAIR OF ANGLE'/,
     1'          BRACKETS   BEFORE   THE   DESCRIPTOR   LIST,  THE'/,
     1'          PERCENTAGE  OF  DATA  ITEMS  FOR  EACH CELL IN AN'/,
     1'          N-WAY TABULATION IS PRINTED.'/,
     1'          EX: <TAB> (YEAR,MONTH,SPCONTS)')
     WRITE(6,4)
4    FORMAT(//,'II. THE BOOLEAN EXPRESSION,  WHICH  DEFINES THE SUBSET OF'/,
     1'     ITEMS FROM THE DATA BANK,  WHICH MUST  BE  RETRIEVED.'/,
     1'     TO COMPLETE THIS,  THE NAME OR CODE OF THE DESCRIPTOR'/,
     1'     AND ITS TYPE SHOULD BE ENTERED, (SEPARATED BY ,).'/,
```

TABLE 5.1. (Continued).

```
     1'      EX: YEAR,80'/,
     1'          MONTH,MAY'//,
     1'      THIS    SUBSET   MAY   CONSIST  OF  A  COMBINATION  OF'/,
     1'      DESCRIPTORS.  IN  THIS  CASE THE  LOGICAL  STATEMENTS'/,
     1'      "AND/OR" AND PARENTHESES  "()"  CAN BE USED TO SELECT'/,
     1'      THE  SUBSET.  IF  "ALL"  IS  ENTERED  THE  SUBSET  OF'/,
     1'      INTEREST IS EQUAL TO THE WHOLE DATA BANK.'/,
     1'      EX: YEAR,80 AND MONTH,MAY'/,
     1'          YEAR,80 OR MONTH,MAY'/,
     1'          (YEAR,80 AND MONTH,MAY) OR DAY,31'/,
     1'          ALL')
C
C
20    ICOUNT=3
      WRITE(6,5)
5     FORMAT(//,'ENTER THE DESCRIPTOR LIST AND THE NECESSARY KEYWORDS.'/,
     1'EX: (YEAR<A>,MONTH<A>,SPCONT<A>)'/,
     1'      (YEAR,SPCONT<T0,T1>)'/,
     1'      <STAT,RESULT> (YEAR,MONTH,SPCONT)'/,
     1'      <TAB> (YEAR,SPCONT>')
      LEN=80
      CALL FREAD(5,'STRING:',LINE(ICOUNT),LEN)
      ICOUNT=ICOUNT+LEN+1
      LINE(ICOUNT)=ASTRIS
      ICOUNT=ICOUNT+1
C
C
      WRITE(6,6)
6     FORMAT(//,'ENTER THE BOOLEAN  EXPRESSION  TO NAME THE',
     1' SUBSET OF'/,'INTEREST FROM THE TOTAL BANK.'/,
     1'EX: YEAR,80 AND MONTH,MAY')
C
C
C
C*******
C*******        TAXIR INTERFACE WITH THE DATA BASE.
C*******
C
      LEN=80
      CALL FREAD(5,'STRING:',LINE(ICOUNT),LEN)
      ICOUNT=ICOUNT+LEN+1
      LINE(ICOUNT)=ASTRIS
C
C
      CALL TAXIR(LINE,ICOUNT)
C
C*******
C*******        CHECK TAXIR MESSAGES.
C*******
C
      CALL ERRCHK(ERRNUM)
      IF(ERRNUM.EQ.0) GO TO 30
      CALL TAXIR('DUMMY',-1)
      CALL CMDNOE('$EM -CHECK',10)
      WRITE(6,7)
7     FORMAT(//,'THE LIST CAN NOT BE PRINTED.'/,
     1'THE STATEMENT YOU HAVE TYPED CONTAINS ERROR(S).'/,
     1'DO YOU WANT TO TRY AGAIN? (Y/N)')
C
C
```

TABLE 5.1. (Continued).

```
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 40
C
C
30    RETURN
      END
      SUBROUTINE TERMIN
C
C
C
C       ************************************************************
C       *                                                          *
C       *       SUBROUTINE TERMIN; TO TERMINATE THE EXECUTION      *
C       *       OF THE PROGRAM.                                    *
C       *                                                          *
C       ************************************************************
C
C
C
      LOGICAL   EQUC
C
C
      WRITE(6,1)
1     FORMAT(//,'DO YOU WANT TO END THE EXECUTION OF THE PROGRAM? (Y/N)')
C
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP))  STOP
C
C
      WRITE(6,2)
2     FORMAT('COMMAND IGNORED.')
C
C
      RETURN
      END
      SUBROUTINE ERRCHK(ERRFLG)
C
C
C
C       ************************************************************
C       *                                                          *
C       *       SUBROUTINE ERRCHK; TO CHECK THE ERROR MESSAGES     *
C       *       PRODUCED BY TAXIR PROGRAM.                         *
C       *                                                          *
C       ************************************************************
C
C
C
      INTEGER ERRFLG
      LOGICAL*1 ERROR(6)
      LOGICAL EQUC
C
C*******
C*******       READ TAXIR MESSAGES FROM THE FILES ASSIGNED TO SERCOM, WHICH
C*******       CONTAINS ERROR MESSAGES RECOGNIZED AT THIS POINT.
C*******
C
      READ(7,1) ERROR
1     FORMAT(6A1)
```

TABLE 5.1. (Continued).

```
      CALL CMDNOE('$EM -CHECK',10)
      ERRFLG=1
      IF(EQUC('0READY',ERROR)) ERRFLG=0
C
C
      RETURN
      END
```

TABLE 5.2. File HELP.

---

```
     **THIS IS AN INTERACTIVE PROGRAM**

IF YOU  ARE NOT NOW  USING THE  "UPPER CASE"
OPTION, PLEASE DO SO. SINCE THIS PROGRAM CAN
WORK ONLY WITH THE "UPPER CASE" OPTION.


THE GOAL OF THIS  PROGRAM  IS TO HELP YOU TO
RETRIEVE THE  INFORMATION YOU  NEED FROM ANY
OF THE 13  EXISTING  DATA  BANKS,  WHICH ARE
LISTED AS FOLLOW:

LAKE.PHYTOPLANKTON                 OR   1
ENTRAINED.PHYTOPLANKTON            OR   2
LAKE.ZOOPLANKTON                   OR   3
ENTRAINED.ZOOPLANKTON              OR   4
LAKE.BENTHOS                       OR   5
ENTRAINED.BENTHOS                  OR   6
IMPINGED.BENTHOS                   OR   7
ADULT.FISH.SUMMARY.STATISTICS      OR   8
LAKE.ADULT.FISH                    OR   9
IMPINGED.ADULT.FISH                OR  10
LAKE.LARVAE                        OR  11
ENTRAINED.LARVAE                   OR  12
NUTRIENT.AND.ANION                 OR  13
LAKEWATER.CHEMISTRY                OR  14
SEDIMENTS                          OR  15

THERE ARE SEVEN COMMANDS IN THIS PROGRAM;

HELP                                 H
SELECT DATA BANK                     S
SELECT OUTPUT OPTION                 O
VARIABLE DESCRIPTION                 V
PERCENTAGE OF TOTAL DATA ITEMS       P
QUERY                                Q
TERMINATE THE PROGRAM                T


HINTS:  FOR DATA  RETRIEVAL, THE SEQUENCE OF
OPERATIONS LISTED BELOW IS USUALLY FOLLOWED;

(1)  FIRST  YOU  SHOULD SELECT THE DATA BASE
     OF INTEREST.
(2)  ONCE THAT IS DONE,  YOU SHOULD TELL THE
     COMPUTER WHAT  DEVICE YOU ARE USING AND
     HOW YOU WANT YOUR INFORMATION  PRINTED.
     THIS CAN BE ACCOMPLISHED BY  TYPING "O"
     AND RESPONDING TO THE  QUESTIONS  WHICH
     APPEAR AFTER THE COMMAND "O".
(3)  THREE TYPES OF  INFORMATION CAN THEN BE
     ACCESSED;   IF YOU  NEED THE  VARIABLES
     CONTAINED IN THE DATA BASE, TYPE "V".
     IF YOU WANT TO  KNOW THE  PERCENTAGE OF
     DATA ITEMS IN THE DATA QUERY TYPE "P".
     IF YOU NEED THE  ACTUAL DATA FOR ALL OR
     PART OF THE DATA BASE, TYPE "Q".

PLEASE ANSWER TO ALL OF THE QUESTIONS, WHICH
APPEAR AFTER YOU TYPE THE COMMAND.
```

TABLE 5.2. (Continued).

(4) FINALLY, WHEN YOU DO NOT WANT TO PROCEED FURTHER, "T" WILL TERMINATE THE RETRIEVAL OPERATIONS AND RETURN YOU TO MTS.

TO GET THE DESCRIPTION OF THE COMMANDS TYPE "H" FOR HELP, THEN TYPE THE FIRST LETTER OF EACH COMMAND WHILE YOU ARE IN HELP MODE, (EX: S, FOR SELECT DATA BANK).

HELP "H"
====

| | |
|---|---|
| FUNCTION: | TO EXPLAIN ONE OR MORE COMMANDS FOR THE USER. |
| HOW TO CALL: | PRESS "H" TO GET A LIST OF ALL THE EXISTING COMMANDS.<br>IF YOU ARE INTERESTED IN A SPECIFIC COMMAND ENTER THE FIRST LETTER OF THAT COMMAND, (H,S,O,V,P,Q,T). |

SELECT DATA BANK "S"
================

| | |
|---|---|
| FUNCTION: | TO SELECT ONE OF THE EXISTING DATA BANKS. |
| HOW TO CALL: | PRESS "S" TO CHOOSE A DATA BANK AND TO GET A LIST OF EXISTING DATA BANKS. |

SELECT OUTPUT OPTION "O"
====================

| | |
|---|---|
| FUNCTION: | TO CHOOSE AN OUTPUT FILE/DEVICE THAT OUTPUTS GET PRINTED ON, (EX: TERMINAL SCREEN, LINE PRINTER, LINE FILE). |
| HOW TO CALL: | PRESS "O" TO SELECT THE OUTPUT FILE/DEVICE. |

TABLE 5.2. (Continued).

| | |
|---|---|
| | **VOCABULARY "V"** |
| FUNCTION: | TO GET THE VOCABULARY LIST FOR SOME OR ALL OF THE DESCRIPTORS. |
| HOW TO CALL: | PRESS "V" TO GET THE VOCABULARY LIST. |
| | **PERCENTAGE "P"** |
| FUNCTION: | TO PROVIDE THE USER WITH THE PERCENTAGE OF ITEMS IN THE TOTAL DATA BANK BELONGING TO THE SUBSET OF INTEREST. |
| HOW TO CALL: | PRESS "P" TO GET THE PERCENTAGE. |
| | **QUERY "Q"** |
| FUNCTION: | TO RETRIEVE INFORMATION FROM THE SELECTED DATA BANK. |
| HOW TO CALL: | PRESS "Q" TO RETRIEVE THE ACTUAL DATA FOR ALL OR PART OF THE SELECTED DATA BASE. |
| | **TERMINATE THE PROGRAM "T"** |
| FUNCTION: | TO TERMINATE THE EXECUTION OF THE PROGRAM. |
| HOW TO CALL: | PRESS "T" TO END THE PROGRAM. |

TABLE 5.3. Program LINK.

```
C RUN PROGRAM LINK RESULTS IN CREATION OF TEMPORARY  FILE -F,
C WHERE "F" INDICATES THE NAME OF THE FILE WHICH CONTAINS THE
C COMMANDS NECESSARY FOR RESTORATION OF THE DATA BASES FROM A
C *FS TAPE.
C UNIT 5 IS ASSIGNED TO *MSOURCE* TO READ THE  CODE NUMBER(S)
C OF THE DATA BANK(S) WHICH NEED TO BE RESTORED.
C UNIT 6 IS  ASSIGNED TO *SINK* TO PRINT THE  MESSAGES ON THE
C TERMINAL.
C UNIT 7 IS ASSIGNED TO FILE "-G" FOR *FS USE.
C UNIT 8 IS  ASSIGNED TO FILE "-F" FOR  RESTORATION COMMANDS.
C
C
C
C      *************************************************
C      *                                               *
C      *        PROGRAM LINK; TO RESTORE THE DATA      *
C      *        BASES FROM THE TAPE.                   *
C      *                                               *
C      *************************************************
C
C
C
      INTEGER IARR(15),NUM,N
      LOGICAL*1 RESP
      LOGICAL EQUC
C
C*******
C*******       ENTER THE CODE NUMBER(S) OF THE DATA BANK(S).
C*******
C
30    WRITE(6,1)
1     FORMAT(//,'ENTER THE CODE NUMBER(S) OF THE DATA BANK(S)  WHICH YOU'/,
     1'LIKE TO WORK WITH DURING THE EXECUTION OF THIS PROGRAM.'/,
     1'THESE DATA BANKS WILL BE RESTORED FROM THE TAPE AND CAN'/,
     1'BE ACCESSED LATER ONE AT A TIME. CODE NUMBERS SHOULD BE'/,
     1'SEPARATED BY A ",". (EX: 1,2,3,4)')
C
      N=FREAD(5,'INTEGER VECTOR:',IARR,15)
      IF(N.NE.0) GO TO 10
      WRITE(6,2)
2     FORMAT(//,'NO ENTRY!'/,
     1'DO YOU NEED THE LIST OF THE DATA BANKS? (Y/N)')
C
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 20
      GO TO 30
C
C*******
C*******       PRINT THE LIST OF DATA BANKS IF NEEDED.
C*******
C
20    WRITE(6,3)
3     FORMAT(//,'THE DATA BANKS AND THEIR CODE NUMBERS ARE;',
     1//,
     1'LAKE.PHYTOPLANKTON             OR    1'/,
     1'ENTRAINED.PHYTOPLANKTON        OR    2'/,
     1'LAKE.ZOOPLANKTON               OR    3'/,
     1'ENTRAINED.ZOOPLANKTON          OR    4'/,
     1'LAKE.BENTHOS                   OR    5'/,
     1'ENTRAINED.BENTHOS              OR    6'/,
```

TABLE 5.3. (Continued).

```
     1'IMPINGED.BENTHOS                   OR    7'/,
     1'ADULT.FISH.SUMMARY.STATISTICS      OR    8'/,
     1'LAKE.ADULT.FISH                    OR    9'/,
     1'IMPINGED.ADULT.FISH                OR   10'/,
     1'LAKE.LARVAE                        OR   11'/,
     1'ENTRAINED.LARVAE                   OR   12'/,
     1'NUTRIENT.AND.ANION                 OR   13'/,
     1'LAKEWATER.CHEMISTRY                OR   14'/,
     1'SEDIMENTS                          OR   15')
C
      GO TO 30
C
C*******
C*******        CHECK THE ENTERED CODE NUMBER(S).
C*******
C
10    J=0
      DO 40 I=1,N
      IF((IARR(I).NE.0).AND.(IARR(I).LT.16)) GO TO 40
      J=J+1
40    CONTINUE
C
      IF(J.EQ.N) GO TO 50
      IF((J.NE.0).AND.(J.LT.N)) GO TO 60
      GO TO 70
C
50    WRITE(6,4)
4     FORMAT(//,'THE ENTERED CODE NUMBER(S) ARE NOT ACCEPTABLE.')
80    WRITE(6,5)
5     FORMAT(//,'DO YOU NEED THE LIST OF THE DATA BANKS? (Y/N)')
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 20
      GO TO 30
C
60    WRITE(6,6) J
6     FORMAT(//,I5,1X,'WRONG CODE NUMBER(S).'/,
     1'DO YOU WISH TO REENTER THE LINE? (Y/N)')
      CALL FREAD(5,'STRING:',RESP,1)
      IF(EQUC('Y',RESP).OR.EQUC('y',RESP)) GO TO 80
C
C*******
C*******        CREATION OF FILE "-G".
C*******
C
70    WRITE(7,7)
7     FORMAT('RESPONDS')
C
      CALL CMDNOE('$EM -G',6)
C
      DO 90 I=1,N
      WRITE(7,8)
8     FORMAT('Y')
90    CONTINUE
C
C*******
C*******        CREATION OF FILE "-F".
C*******
C
      WRITE(8,9)
9     FORMAT('COMMANDS')
```

TABLE 5.3. (Continued).

```
C
      CALL CMDNOE('$EM -F',6)
C
      DO 100 I=1,N
      NUM=IARR(I)
      GO TO (120,130,140,150,160,170,180,190,210,220,230,240,250,260,270),NUM
C
      WRITE(6,11) NUM
11    FORMAT(//,I5,1X,'NOT AN ACCEPTABLE CODE NUMBER.')
      GO TO 100
C
120   WRITE(8,12)
12    FORMAT('RESTORE LAKE.PHYTOPLANKTON -LAKE.PHYTO')
      GO TO 100
C
130   WRITE(8,13)
13    FORMAT('RESTORE ENTRAINED.PHYTOPLANKTON -ENT.PHYTO')
      GO TO 100
C
140   WRITE(8,14)
14    FORMAT('RESTORE LAKE.ZOOPLANKTON -LAKE.ZOO')
      GO TO 100
C
150   WRITE(8,15)
15    FORMAT('RESTORE ENTRAINED.ZOOPLANKTON -ENT.ZOO')
      GO TO 100
C
160   WRITE(8,16)
16    FORMAT('RESTORE LAKE.BENTHOS -LAKE.BEN')
      GO TO 100
C
170   WRITE(8,17)
17    FORMAT('RESTORE ENTRAINED.BENTHOS -ENT.BEN')
      GO TO 100
C
180   WRITE(8,18)
18    FORMAT('RESTORE IMPINGED.BENTHOS -IMP.BEN')
      GO TO 100
C
190   WRITE(8,19)
19    FORMAT('RESTORE ADULT.FISH.SUMMARY.STATISTICS -AD.FISH.S.S')
      GO TO 100
C
210   WRITE(8,21)
21    FORMAT('RESTORE LAKE.ADULT.FISH -L.AD.FISH')
      GO TO 100
C
220   WRITE(8,22)
22    FORMAT('RESTORE IMPINGED.ADULT.FISH -I.AD.FISH')
      GO TO 100
C
230   WRITE(8,23)
23    FORMAT('RESTORE LAKE.LARVAE -L.LARVAE')
      GO TO 100
C
240   WRITE(8,24)
24    FORMAT('RESTORE ENTRAINED.LARVAE -E.LARVAE')
      GO TO 100
C
250   WRITE(8,25)
```

TABLE 5.3. (Continued).

```
25    FORMAT('RESTORE NUTRIENTS -NUTRIENTS')
      GO TO 100
C
260   WRITE(8,26)
26    FORMAT('RESTORE LAKEWATER -LAKEWATER')
      GO TO 100
C
270   WRITE(8,27)
27    FORMAT('RESTORE SEDIMENTS -SEDIMENTS')
100   CONTINUE
C
C*******
C*******          END OF THE *FS COMMANDS IN FILE "-F".
C*******
C
      WRITE(8,28)
28    FORMAT('STOP')
C
      WRITE(6,29)
29    FORMAT(//,'AT THIS POINT THE TAPE IS BEING MOUNTED AND THE REQUESTED'/,
     1'DATA BANK(S)  ARE   BEING  RESTORED.  COMPLETION OF THIS'/,
     1'PROCESS WILL TAKE A FEW MINUTES. PLEASE STAND BY!')
C
C
      STOP
      END
```

and then passing their responses on to the Taxir program. This program is made possible by the fact that Taxir can be loaded and run as a subroutine of a larger program. Using this feature in programming, two programs can be loaded together, and control commands can be passed to each other without unloading the first program. This allows both programs to remain loaded and maintain an updated value of their variables. However, if a program is loaded along with the initialization of the variables by using an ordinary run command, problems can occur when a second run command is issued. This results in loading the new program and unloading of the first program, and can lead to the loss of the updated values for the parameters of the first program.

The program INTERACT, using a Taxir FORTRAN callable subroutine, stores the necessary commands (statements) in an array of characters and passes them to Taxir as an argument of the subroutine, as shown in the following form:

```
CALL TAXIR (ARRAY,LENGTH)
```

ARRAY is the first calling argument containing the Taxir command, and LENGTH is the 2nd calling argument representing the length in characters of the command. The LENGTH argument is important because at each call Taxir should look only at the meaningful characters stored in the array at that time so as to avoid problems which can be caused by the characters left by the previous commands of greater length. As an example, the Taxir call (as seen in various uses in program INTERACT) can be:

```
CALL TAXIR ('SET OUTPUT=*MSINK*',18)
```

Here, 18 is the number of the characters in this command, and this call could be made when the terminal screen is to be selected as the output device.

The above explanations show how, in general, different Taxir commands are passed by INTERACT (or any other FORTRAN program) to Taxir. However, Taxir can be called from INTERACT in two other special forms for two specific purposes. On these two calls, the LENGTH arguments have definite values, but the ARRAY arguments can be ignored and replaced by dummy parameters. These two calls are explained as follows:

1. LENGTH=0 CALL TAXIR ('DUMMY',0)

   This call initializes Taxir and is equivalent to invoking Taxir with the $RUN *TAXIR command. This call is made only once in the beginning of the program INTERACT.

2. LENGTH=-1 CALL TAXIR ('DUMMY',-1)

   This is equivalent to pressing the return key (in case Taxir is running by itself) to cancel a statement or a data item. This call is used in INTERACT whenever an error in the statement (command) passed to Taxir is detected.

PROGRAM LINK

There are 15 different data banks contained in the Cook data base. Some of these data banks are very large and occupy a large memory space. As a result, it is not economical to keep all the data on-line. Our approach is to save these data banks on magnetic tape and to restore them in temporary files when they are needed.

The program LINK (shown in Table 5.3) is designed to accomplish this task. This program can be called by INTERACT whenever the request is made

to restore some or all of the data base. When users respond to the computer query as to the names of data banks to be restored, the restoration of these data banks will be made in temporary files. This process, however, often takes considerable time, especially during peak computer use periods.

## PROCEDURES FOR OPERATING INTERACT AND LINK

To run the program INTERACT, the user should first sign on to MTS. This is done by entering the computer center account ID (CCID) and the password for that account shown as follows:

```
#SIGNON XXXX (account ID)
?XXXX        (Password)
```

Then enter the command:

```
#SOURCE INTERACTIVE
```

This statement will start the execution of the program INTERACT. The communication from this point on is interactive; the users simply respond to the questions asked by the program. To ensure the success of the data retrieval, it is important to respond to all the questions asked by the program. However, during the execution of the program if there is a need to select a data bank, which may or may not be on-line, the program will first check whether the data bank being requested is on-line. If the file is not on-line, the program LINK will be run automatically to restore part or all of the requested data.
The flow-diagram in Figure 5.1 represents the major steps in the above operations.

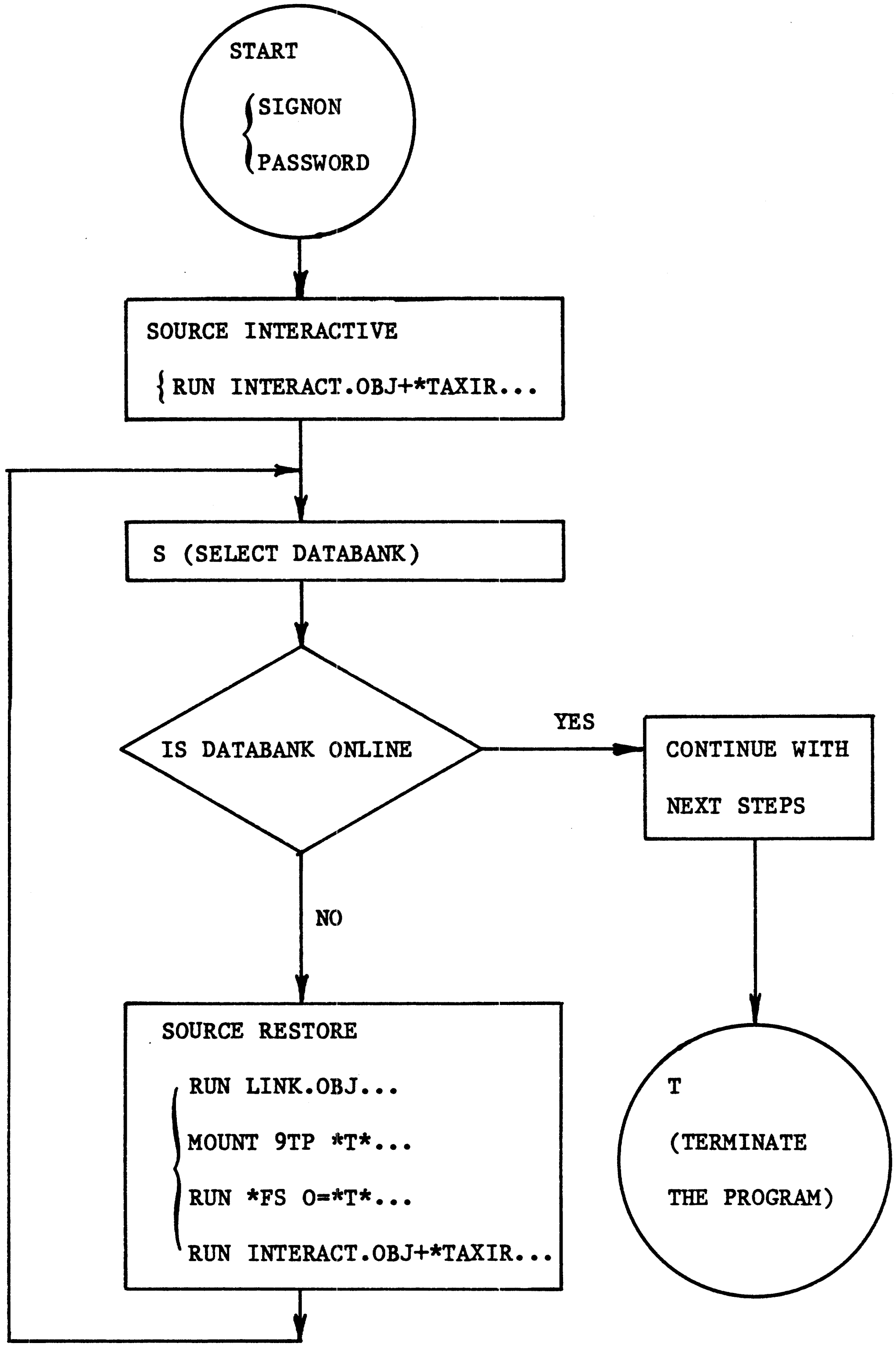



Figure 5.1. Flow-chart representation of stages involved in operations of INTERACT and LINK.

UNIT ASSIGNMENTS FOR PROGRAMS INTERACT AND LINK

Any CCID which is used to run the interactive program will have access to the following four files:

File I: INTERACT.OBJ

This file contains the object codes for the INTERACT program.

File II: LINK.OBJ

The object codes for the LINK program are in this file.

File III: INTERACTIVE

This file consists of the following RUN command:

```
RUN INTERACT.OBJ+*TAXIR 5=*MSOURCE* 6=*SINK* 7=-CHECK
SERCOM=-CHECK
```

This run command invokes Taxir and INTERACT simultaneously. Units 5 and 6 are assigned to the terminal screen to serve as interactive input/output devices. Taxir error messages are written on a device called SERCOM. SERCOM and Unit 7 are assigned to the temporary file -CHECK. This file is used to detect and control typing errors made by the user.

File IV: RESTORE

There are four commands in this file:

Command A:

```
RUN LINK.OBJ  5=*MSOURCE*  6=*SINK*  7=-G  8=-F
```

Unit 8 is assigned to file -F. File -F contains the restoration commands. Units 5 and 6 are assigned to the terminal screen as explained in III. Unit 7 is assigned to -G to be used later in MTS file saving program (*FS).

Command B:

MOUNT C0073A 9TP *T* VOL=COOK 'COOK'

This line of file RESTORE specifies that the 9-track computer tape should be mounted.

Command C:

RUN *FS 0=*T* SCARDS=-F GUSER=-G SERCOM=-S SPRINT=-SP

Running the *FS program will result in restoration of the requested databanks. SCARDS is assigned to -F for input commands. GUSER is assigned to -G which is created by Command A for the source of responses to *FS prompting messages. SERCOM is assigned to -S for the error messages and SPRINT is assigned to -SP for saving other messages from *FS.

Command D:

RUN INTERACT.OBJ+*TAXIR 5=*MSOURCE* 6=*SINK* 7=-CHECK SERCOM=-CHECK

Finally, this last line of file RESTORE is the same run command as in Command A. This identical run command will restart the program INTERACT. At this point, the file restoration is complete and data retrieval can be continued.

# CHAPTER 6

# DISCUSSION

This report documents in detail the procedures used and the programs written in establishing the data base management system for the D. C. Cook ecological study and describes the data contained in the data base as well as the way in which these data can be accessed. This documentation can be used as a reference when accessing the data base and to clarify questions that arise during the use of this system.

A computer data base management system is essential for a large project because it can increase efficiency in report writing, data analyses, and data interpretation. It can also enhance data exchange and utilization; provide an efficient way to archive the data; and act as a safeguard for access to and centralized management of the data.

The Cook computer data base encompasses 15 individual data banks. The total includes more than one-half million cases of biological, chemical, and physical information on the nearshore of southeastern Lake Michigan. This data base is considered one of the largest water quality information bases for the southeastern portion of the lake. Considerable experience has been gained from the establishment of this data base. This experience can be useful for establishing a computer data system for other projects and is discussed in the paragraphs which follow.

Any sizable research project should have a computerized data base management system. Such a system should be established at the onset of the research project, beginning at the same time as the beginning of data collection. When one is still in the research planning phase, considerations should be given to

1) the type of data which will be collected, 2) the kind of analysis and report format which will be needed, 3) the type of access which is expected, and 4) the type of information retrieval which will be used frequently. These considerations are critical to ensure that the computerized data base meets the needs of the users. Once the answers for these questions are provided, a uniform data reporting format should be used. This can reduce the time and effort needed to establish a data base management system.

One needs to know what DBMS programs are available from the main frame of the computer and at what cost for data entry and subsequently updating. One should also know the capabilities and limitations of the available DBMS programs, whether the capabilities of a DBMS program meet the basic needs for a project, and whether the limitations would impose a serious restriction on the operation and use of the data base. Considerations of DBMS should also cover the maintenance and expansion of a data base and the degree of difficulty for a person to use and operate the data base management system.

The discussion thus far has assumed that the state of computer technology and data base management has remained unchanged. It is important to realize that is not the case, that a DBMS which serves you today will certainly differ greatly at some time in the future. We have observed rapid progress in the use and operation of computers in the past 10 years. We can expect that similar even faster progress may be made in the time to come. For example, it would not be unthinkable that one could simply speak to a computer receiver to retrieve needed information from a DBMS in the near future, as voice recognition and natural language queries become components of data base management systems. As computers become more powerful and able to store large amounts of data less expensively, a DBMS could include graphics or instant results of statistical

analyses with a few simple commands. Simplified protocol can make the use of DBMS an easy task and will enable the user to access many on-line information services without learning specialized techniques or command languages for each one. Accessing an ecological data base in this case would not differ too much from getting cash from your bank account by using a computer terminal. By then, DBMS will truly be an integral part of our daily life. The tedious task of data selection and analyses for aquatic studies can be accomplished in a matter of a few minutes.

While dreaming of possible future directions in computer technology for data base management systems, we hope and believe that the Cook data base management system represents the state of the art for the present and can facilitate the use and access of ecological data for southeastern Lake Michigan.

## BIBLIOGRAPHY


Ayers, J. C., and E. Seibel. 1973. Benton Harbor Power Plant Limnological Studies. Part XVII. Program of aquatic studies related to the Donald C. Cook Nuclear Plant. Univ. of Michigan, Great Lakes Res. Div., Spec. Rep. No. 44, ii, 1, 2 pp.

Bassler, R. A., and J. J. Logan. 1976. The technology of the data base management systems. College Readings, Inc., Virginia.

Berryman, J. 1981. Data base management. Computing Center Newsletter. Univ. of Michigan, Computing Center. Vol. 11, Nos. 10, 11, 12.

Bridges, T. 1982. Data base machines. J. Data Management 11:14-16.

Brill, B. C. 1983. TAXIR Primer Manual. Univ. of Michigan, Computing Center.

Cordenas, A. F. 1979. Data base management systems. Allyn and Bacon, Inc., Massachusetts.

Enger, N. L. 1983. Developing data base structure specifications. J. Data Management 2:16-19.

Hamper, R. 1983. Integrating WP and DP. Data Processing J. London 1:17-18.

Hermann, K. H. 1983. Caught between two stools. Data Processing J. London 1:11-13.

Kahn, M. A., D. L. Rumelhart, and B. L. Bronson. 1977. MICRO Manual. Univ. of Michigan and Wayne State Univ., ILIR.

Kaplowitz, H. 1981. Application development in a data base environment. J. Data Management 9:24-26.

Lane, L. L. 1980. First evaluate user needs, limits, then product assets, limits. J. Data Management 5:52-55.

Martin, J. 1977. Data base organization. Prentice-Hall Inc., New Jersey.

Omar, M. H. 1980. DBMS simplified. J. Data Management 10:23-26.

Russell, J. C. 1983. All the info-all the time-on line. J. Data Management 2:41-42.

Schussell, G. 1983. Mapping out the DBMS territory. J. Data Management 2:24-27.

Silbey, V. 1979. Documentation standardization. J. Data Management 4:32-35.

_____. 1976. SPIRES Manual. Stanford Univ., Stanford, California.